Algorithmes d'optimisation pour la résolution du problème de stockage de conteneurs dans un terminal portuaire

Ndèye Fatma Ndiaye

Université du Havre | École doctorale SPMII

Thèse

présentée pour obtenir le titre de DOCTEUR en
Mathématiques Appliquées Informatique

Algorithmes d'Optimisation pour la Résolution du Problème de Stockage de Conteneurs dans un Terminal Portuaire

Ndèye Fatma NDIAYE épouse DIAGNE
farlou@live.fr

Soutenue publiquement le 23 Juin 2015 devant un jury composé de :

Rapporteurs	Professeur Abderrafiaa KOUKAM	Université de Technologie de Belfort-Montbéliard
	Professeur Ali Ridha MAHJOUB	Université de Paris Dauphine
Examinateurs	Professeur Abdelkader SBIHI	École de Management de Normandie
	Professeur Cyril FONLUPT	Université du Littoral Côte d'Opale
Directeur de thèse	Professeur Adnan YASSINE	Université du Havre
Co-encadrant de thèse	Docteur Ibrahima DIARRASSOUBA	Université du Havre

Laboratoire de Mathématiques Appliquées du Havre (LMAH)

Thèse réalisée au Laboratoire de Mathématiques Appliquées du Havre (LMAH)
Université du Havre
25 rue Philippe Lebon, BP 1123
76063 Le Havre Cedex, France
Tél : +33 (0)2 32 74 49 16
Fax : +33 (0)2 32 74 49 11
Web : http ://lmah.univ-lehavre.fr/

Sous la direction de Adnan YASSINE adnan.yassine@univ-lehavre.fr

Co-encadrement Ibrahima DIARRASSOUBA diarrasi@univ-lehavre.fr

Financement Bourse du gouvernement Sénégalais

Résumé

Un terminal à conteneurs constitue une interface inter-modale essentielle pour le réseau de transport mondial. La compétitivité d'un port est mesurée par l'efficacité de ses terminaux. Cette dernière est principalement reflétée par la quantité de conteneurs chargés et déchargés. Une manutention efficace des conteneurs dans les terminaux portuaires est d'une importance cruciale pour la réduction des coûts de stockage, de transport, de gestion et pour la détermination des plans d'embarquement.

Le problème de stockage de conteneurs représente un des principaux problèmes surtout avec la croissance considérable du nombre de conteneurs passant par les ports maritimes et la limitation des zones de stockage dans ces ports. Il a un impact considérable sur l'efficacité des autres opérations portuaires. L'objectif visé dans la résolution de ce problème est de trouver un plan de stockage optimal qui précise l'emplacement de stockage idéal pour chaque conteneur et qui tient compte des contraintes réelles de stockage.

Dans cette thèse, nous traitons le problème de stockage de conteneurs dans un terminal portuaire, en distinguant deux cas :

- le cas statique qui consiste à déterminer un plan de stockage optimal,
- le cas dynamique qui prend en considération les variations de l'état de la cour de stockage.

Dans un premier temps, nous présentons un état de l'art et une étude bibliographique dans laquelle sont relatés les travaux déjà réalisés dans ce domaine. Après cela, nous abordons successivement les deux cas de stockage. Pour chacun d'entre eux, nous présentons une étude analytique approfondie, une modélisation mathématique efficace qui reflète le cas réel et nous développons des méthodes de résolution numériques adéquates qui englobent des algorithmes de résolution efficaces et robustes. Pour effectuer une étude théorique approfondie sur ce problème, nous proposons une démonstration de sa complexité, qui réaffirme son caractère NP-difficile (NP-complet). Ce genre de problème, et dans le cas général où la dimension est très grande, peut être difficilement résolu avec des méthodes exactes. Les simulations numériques réalisées avec le logiciel d'optimisation "ILOG CPLEX" le confirment, raison pour laquelle nous proposons un algorithme de Branch-and-Cut, qui est une méthode de résolution exacte et qui nous permet de repousser les limites de "ILOG CPLEX". Cependant, cet algorithme a aussi des limites car il est très gourmand en place mémoire, en plus, il nécessite un temps d'exécution considérable lorsque le nombre de conteneurs à stocker devient très grand. Ainsi, nous proposons des algorithmes méta-heuristiques, à savoir : un algorithme de colonie d'abeilles, un algorithme génétique, un algorithme de colonie de fourmis, et un algorithme de recuit simulé. Ces algorithmes sont des méthodes de résolution approchées, c'est-à-dire que l'optimalité des solutions qu'ils fournissent n'est pas assurée, mais des comparaisons avec les résultats

obtenus par "ILOG CPLEX", réalisés sur des instances de tailles raisonnables, ont prouvé la bonne qualité des solutions fournies par ces algorithmes. Afin de combiner leurs performances et d'éviter la convergence rapide vers un optimum local, des hybridations entre ces différents algorithmes méta-heuristiques ont été proposées. Ces algorithmes hybrides ont prouvé leur efficacité par la bonne qualité de leurs solutions. La première hybridation est un renforcement de l'algorithme de colonie de fourmis par un algorithme génétique, tandis que les deux autres représentent des recherches locales à chaque itération effectuées par l'algorithme du recuit simulé dans l'algorithme de colonie de fourmis d'une part, et l'algorithme génétique d'autre part. Ces hybridations ont permis d'accélérer la convergence des algorithmes et parfois d'améliorer la qualité des meilleures solutions obtenues.

Mots clés :

Optimisation Combinatoire, Modélisation Mathématique, Méthodes Exactes, Méthodes Méta-heuristiques, Stockage de Conteneurs.

Remerciements

À mes parents, qui se sont dévoués corps et âmes pour la réussite de leurs enfants.

Je remercie les membres du jury, ainsi que tous les enseignants qui ont participé à ma formation notamment Monsieur Adnan Yassine et Monsieur Ibrahima Diarrassouba, sans oublier mes collègues du laboratoire de mathématiques appliquées du Havre (LMAH) particulièrement Madame Zeinebou Zouber.

Un grand remerciement à mon mari qui m'a toujours soutenu.

Mes pensées et ma gratitude vont également vers Monsieur Seydou Touré et Monsieur Babacar Diop qui ont été de solides appuis.

Un grand MERCI au Sénégal pour avoir financé ma formation parmi tant d'autres.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Cette introduction a pour objectif de présenter le contexte du travail réalisé dans cette thèse, montrer l'importance de la gestion d'un terminal à conteneurs portuaire pour l'amélioration du transfert de marchandises dans une chaîne logistique globale de bout en bout, définir notre problématique de recherche qui représente un élément essentiel, parmi d'autres, de la performance de cette chaîne logistique, modéliser ce problème sous forme d'un programme mathématique et enfin présenter les méthodes numériques efficaces proposées pour sa résolution.

Cadre général

La conteneurisation des marchandises a joué un rôle important dans le développement des réseaux internationaux de transport inter-modal. En effet, avec l'arrivée du conteneur durant les années 50, un important réseau international de transport basé sur l'utilisation d'un format standard de cargaison s'est développé. Cette standardisation a permis d'accélérer le transfert de la marchandise d'un mode de transport à un autre mais en même temps elle a rendu les contrôles manuels quasiment impossibles. Par conséquent, des efforts supplémentaires se sont imposés pour la création de moyens de contrôles techniques. Aujourd'hui, les opérations de transfert d'un mode de transport à un autre demeurent l'élément clé d'un système de transport performant. Parmi ces points de transfert, les terminaux portuaires de conteneurs sont généralement identifiés comme un maillon important de la chaîne logistique globale de bout en bout.

Les terminaux à conteneurs constituent des interfaces inter-modales essentielles pour le réseau de transport mondial. Ils jouent un rôle important dans la logistique moderne comme un centre de transport maritime. La compétitivité d'un terminal est principalement reflétée par son efficacité de transbordement à cause des charges payées par un bateau qui dépendent du temps de rotation et du nombre de conteneurs chargés et déchargés à un terminal. Le terminal à conteneurs est généralement exploité avec trois types d'équipements :

- des grues de quai qui assurent le chargement et le déchargement des navires,
- des camions de cour (véhicules auto guidés, cavaliers, etc.) qui assurent le transport des conteneurs du quai vers les zones de stockage et vice versa,
- des grues de cour qui assurent le stockage des conteneurs dans les zones de stockage.

Suite à l'accroissement considérable du trafic de conteneurs par voies maritimes, les opérations de manutention dans les terminaux à conteneurs ont fortement augmenté ces dernières années. En effet, une manutention efficace des conteneurs dans des terminaux

est d'une importance cruciale pour la réduction des coûts de transport et la détermination des plans d'embarquement. De ce fait, il est nécessaire de construire de nouveaux terminaux portuaires ou au moins d'améliorer les capacités des anciens. De plus, les infrastructures des terminaux existants vont également s'accroître. Le problème qui se pose est de savoir si les terminaux existants sont assez performants pour manutentionner des flux plus importants de conteneurs ou si les équipements existants doivent être remplacés par d'autres plus efficaces. Dans le but de répondre à ces questions, Il existe de nombreuses études sur l'optimisation des opérations dans un terminal à conteneurs utilisant des nouvelles techniques de la recherche opérationnelle. Ces études portent généralement sur des problèmes spécifiques tels que l'ordonnancement d'un type d'équipement, l'affectation de navires aux quais ou la gestion des espaces de stockage dans le but d'optimiser au maximum le trafic de conteneurs.

Les problèmes opérationnels des terminaux à conteneurs ont été divisés en plusieurs sous problèmes, comme l'assignation des navires, planification d'arrivage, planification des grues de quai, la planification des cavaliers de cour, planification des grues de cour et l'assignation de stockage des conteneurs.

L'organisation de la zone de stockage de conteneurs du terminal requiert une attention particulière. En effet, le respect des contraintes de temps de chargement des navires, des camions et des trains dépend fortement de l'emplacement des conteneurs dans le terminal. Les temps de recherche des conteneurs sont parfois considérables et entraînent des retards importants provoquant des pénalités financières pour les sociétés gérant les terminaux. Afin de pouvoir gérer efficacement cette zone, il est nécessaire de connaître l'emplacement optimal de chaque conteneur. Nos travaux de recherche dans cette thèse traitent précisément ce problème.

La quantité de conteneurs utilisés annuellement dans le commerce maritime international ne cesse d'augmenter. Le flux de conteneurs à travers le monde avoisinait les 601 millions Équivalent Vingt Pieds en 2012 alors qu'il était égal à 35 millions d'Équivalent Vingt Pieds en 1980, [66].

Depuis leur introduction dans le commerce maritime aux environs des années 1960, les conteneurs représentent un concept de standardisation très utile. En effet, ils facilitent les transferts de marchandises entre différentes parties de la chaîne logistique, à savoir : les usines de fabrication, les transitaires, les compagnies maritimes et les clients (intermédiaires et finaux). Les terminaux à conteneurs servent principalement d'interfaces entre différents modes de transport tels que des trains, des camions, des barges et des navires.

L'accroissement mondial des activités des compagnies industrielles entraine l'augmentation des flux de conteneurs. En effet, les besoins d'exportation et d'importation de produits finis et de matières premières évoluent en fonction des usines de fabrication. Ainsi, l'implantation progressive d'entreprises mondiales en Asie rime avec l'augmentation continuelle du nombre de conteneurs qui transitent entre ce continent et les autres. D'ailleurs, selon Wang [110], le nombre de conteneurs qui ont transité entre l'Asie et l'Europe a doublé entre 1990 et 1996, alors que le flux de conteneurs entre l'Europe et les Amériques n'a augmenté que de 10% durant la même période.

Parmi les éléments qui ont contribué au succès du transport maritime de conteneurs, on peut citer :

- Le fait que vers les années 1961, une quantité importante du fret qui circulait sur les routes de la côte Est des États Unis d'Amérique a été conteneurisée, puisque

les premières entreprises qui ont proposé des services de transport maritime de conteneurs ont commencé par assurer le transfert de conteneurs entre ces routes et les ports de l'Amérique central et de l'Amérique du Nord.

- La création continuelle de navires porte-conteneurs qui ont des capacités de plus en plus grandes (allant jusqu'à 10 milles Équivalent Vingt Pieds par navire) facilite le transfert de conteneurs en diminuant les coûts de transport, et par conséquent entraine une augmentation considérable des flux de conteneurs dans les ports et la nécessité d'une gestion optimale des terminaux (matériels, organisation, ordonnancement, etc.).
- Entre les années 1990 et 2003, l'accroissement de la production industrielle mondiale, qui a augmenté de près de 50%, s'est répercuté sur le flux de conteneurs qui a triplé de volume durant la même période.

Pour une meilleure compréhension de ces facteurs de développement du transport maritime de conteneurs, des interprétations et des explications bien détaillées se trouvent dans les références [110] et [92].

Ce développement du transport maritime de conteneurs a suscité l'agrandissement de certains ports et la création des nouveaux. À cela, s'ajoute la nécessité d'investissements en équipements de manutention, de transport, et d'inspection de conteneurs. Selon Günter et Kim [43], la création d'un terminal à conteneurs en eau profonde peut couter jusqu'à un milliard d'euros. Ainsi, pour mieux rentabiliser leurs activités, certains terminaux à conteneurs ont tendance à économiser sur la main d'œuvre, en utilisant des équipements automatisés. Ce phénomène est plus fréquent dans les pays où la main d'œuvre est très chère.

Les taux de croissance élevés des routes maritimes de conteneurs entrainent une compétition considérable entre les terminaux à conteneurs. Cette compétition pousse non seulement les ports à augmenter leurs capacités de manutention, mais aussi à utiliser des logiciels de logistique, des configurations et des technologies modernes, pour faire croître de plus en plus leurs productivités. On peut citer l'exemple du port de Singapour, où le nombre de conteneurs manipulés par employé a quintuplé entre 1987 et 2011 (voir [43]).

D'une manière générale, un terminal à conteneurs est un système complexe qui englobe des inter-actions hautement dynamiques entre des informations parfois incomplètes sur des évènements futures et différents équipements de manutention, de transport, et de stockage. Ainsi, ils existent plusieurs problèmes de décision concernant les terminaux à conteneurs maritimes. Ils peuvent être regroupés en deux catégories : les problèmes de conception et les problèmes de planification. Les résolutions de problèmes de conception se font généralement avant la construction d'un terminal à conteneurs. Ces problèmes doivent être analysés aussi bien sur le plan économique que sur le plan technique. Les plus pertinents d'entre eux sont relatés ci-dessous. Cependant, des informations supplémentaires sont disponibles dans [92].

- Des interfaces multi-modales : Contrairement à leurs homologues asiatiques, la plupart des terminaux à conteneurs européens ont des installations multi-modales ; c'est-à-dire qu'ils sont reliés à des chemins de fer, à des routes, et à des systèmes de navigations. L'intégration de ces différents modes de transport a un impact majeur sur la conception entière du terminal.

- Plan du terminal : La cour de stockage, les chemins de transport, et les quais, représentent des éléments clés pour chaque terminal à conteneurs. Leurs capacités et leurs dispositions spatiales affectent la performance de la configuration du terminal. La détermination du plan d'un terminal à conteneurs concerne aussi la réservation de certains espaces spécialement aménagés pour certains types de conteneurs, tels que des conteneurs frigorifiques, à matières dangereuses, vides, etc.
- Sélection des équipements : différents types d'équipement peuvent être utilisés pour les opérations de manutention et de transport à l'intérieur d'un terminal à conteneurs. Ils diffèrent principalement par leurs degrés d'automatisation et par leurs performances. L'utilisation de grues de stockage automatisées et de véhicules auto guidés devient de plus en plus fréquent, même si ces types d'équipement soulèvent des problèmes de contrôles logistiques complexes.
- Capacité d'amarrage : Elle concerne le nombre de navires qui peuvent être servis ainsi que leurs dimensions, mais aussi la quantité d'emplacements requis dans la cour de stockage et le nombre de véhicules de transfert nécessaire.
- Les systèmes d'information technologiques et les logiciels de contrôle : Un contrôle logistique sur un très grand terminal à conteneurs est un travail extrêmement complexe, qui nécessite des décisions en temps réel sur les affectations d'équipements adéquats aux différentes tâches, et aussi sur la fourniture d'informations détaillées concernant chaque conteneur. L'utilisation de différentes sortes de logiciel et de supports d'information technologiques, de même que des outils d'optimisation sophistiqués, sont des options très pertinentes. Les problèmes de panification concernent différentes opérations dans un terminal à conteneurs. Cependant, une planification décentralisée est nécessaire puisque le système de contrôle logistique d'un terminal à conteneurs est sub-divisé en divers modules pour différents types ou groupes de ressources. Ainsi, on distingue des problèmes de planification et d'ordonnancement spécifiques à l'utilisation de ressources clés.
- L'allocation des postes d'amarrage : Avant l'arrivée d'un navire, un emplacement d'amarrage doit lui être attribué, en tenant compte de sa taille et de la durée prévue de son séjour au port. Néanmoins, des contraintes additionnelles en rapport avec la disponibilité des grues et d'un poste d'amarrage peuvent s'imposer.
- L'allocation des grues : Pour charger ou décharger un grand navire porte-conteneurs, il est nécessaire d'utiliser des grues. Dans un premier temps, les grues sont affectées aux navires, en tenant compte de leurs possibilités d'accéder aux postes d'amarrage. Ensuite, les grues qui sont affectées à chaque navire sont réparties, en affectant chacune à une partie précise du navire, de telle sorte qu'elles ne se gênent pas mutuellement.
- Planification et séquençage d'arrimage : Pour chaque navire, un plan qui précise les emplacements qui sont prévus pour chaque catégorie (destinations, poids, taille, nature, date de départ, etc.) de conteneur est fourni par la compagnie maritime concernée. En se basant sur ce plan, les agents du terminal à conteneurs déterminent l'emplacement de stockage qui est alloué à chaque conteneur sur le navire. Cette dernière affectation a un impact majeur sur le séquençage des chargements et des déchargements des conteneurs, car les planificateurs du terminal se basent sur elle pour établir ces séquences. Pour les conteneurs sortants, en plus de la séquence de

chargement, l'emplacement du navire dans lequel va être stocké chacun d'eux doit être aussi déterminé. Les séquences de chargement et de déchargement sont des données capitales pour l'ordonnancement des grues et des véhicules de transfert.

- Stratégies de stockage et d'empilement : Les grands terminaux à conteneurs d'Europe sont capables de stocker chacun plus de 10 milles conteneurs simultanément, [43]. Puisque la durée moyenne de séjour d'un conteneur dans un port est comprise entre 3 et 5 jours, il est nécessaire de bien gérer la cour de stockage. Cette dernière est composée de plusieurs blocs, qui contiennent chacun des travées, des rangées, et des étages. Les stratégies adoptées pour allouer des emplacements de stockage aux conteneurs ont pour objectif principal de permettre des opérations de stockage et de retrait qui sont rapides tout en évitant autant que possible de provoquer des remaniements. L'efficacité d'une stratégie de stockage dépend de plusieurs facteurs tels que : la configuration de la cour de stockage, les équipements utilisés, les durées de séjour des conteneurs, etc.
- Gestion de la main d'œuvre : L'affectation des tâches aux employés qui manipulent les équipements de manutention et de transfert doit être faite avant le début des opérations.

Dans cette thèse, nous nous intéressons au problème de stockage de conteneurs. Ce dernier consiste à déterminer des emplacements de stockage adéquats pour les conteneurs qui arrivent dans un terminal et qui y séjournent temporairement. Une résolution efficace de ce problème nécessite la prise en considération des contraintes réelles du terminal, telles que le type d'équipement de transfert et de manutention utilisé ainsi que la configuration de la cour de stockage. Outres ces contraintes, il est aussi important d'assurer l'accessibilité de chaque conteneur au moment de sa date de départ, car des opérations supplémentaires peuvent être nécessaires pour déplacer les objets encombrants si tel n'est pas le cas. Cette contrainte n'est pas rigoureusement traité dans la plupart des travaux rencontrés dans la littérature, qui proposent des stockages par groupe, en affectant des groupes de conteneurs à des groupes d'emplacements de stockage. Cette méthode peut fonctionner dans le cas des conteneurs qui ont les mêmes dates de départ et qui partent par le même train ou bien le même navire, mais elle ne convient pas aux conteneurs qui partent par camion. L'un des rares travaux qui ont traité la contrainte d'accessibilité des conteneurs tout en déterminant un emplacement de stockage précis pour chacun d'eux a proposé de les placer dans chaque pile suivant l'ordre décroissant de leurs dates de départ et d'interdire toute sorte de réarrangement, [79]. Cette solution ne convient pas aux terminaux à conteneurs qui reçoivent simultanément des quantités très importantes de conteneurs et qui ont des surfaces de stockage limitées. Raison pour laquelle, nous proposons des méthodes de résolution du problème de stockage de conteneurs qui déterminent un emplacement de stockage précis pour chaque conteneur tout en minimisant les remaniements (voire même les éliminer complètement dans le cas où la grandeur de la cour de stockage le permet). En plus de cela, nous minimisons aussi la distance totale à parcourir par les véhicules de transfert et nous regroupons les conteneurs qui ont des caractères communs. Deux cas de stockage sont distingués dans cette thèse, à savoir : le cas statique et le cas dynamique. Dans le premier, on considère que tous les conteneurs sont arrivés au terminal avant le début des opérations de stockage ; alors que dans le second, on prend en considération les conteneurs qui ne sont pas encore arrivés au terminal au début des opérations de stockage mais qui vont arriver durant l'horizon de planification. Pour

chacun de ces cas, nous proposons une étude analytique suivie d'une modélisation mathématique et des algorithmes de résolution numériques comprenant une méthode exacte (Branch-and-cut), des méthodes méta-heuristiques (colonie d'abeilles, colonie de fourmis, recuit simulé, génétique), et des hybridations entre différentes méta-heuristiques.

Contribution

Depuis son introduction dans le transport maritime, le conteneur est devenu un élément incontournable dans les activités portuaires. Il comporte de nombreux avantages liés à la standardisation de ses dimensions, qui a permis de faciliter ses transbordements entre différents modes de transport. Ainsi, il favorise les gains de temps grâce à l'automatisation des opérations de chargement et de déchargement, en plus de cela, il renforce la sécurité des marchandises en réduisant les risques de rupture de charge lors des transbordements.

Malgré ses nombreux avantages, l'utilisation de conteneurs dans le transport maritime n'a pas manqué de susciter des dépenses colossales d'argent. Cela se justifie par la nécessité d'investissements en espaces de stockage de conteneurs, en équipements et à leurs entretiens. Outres ces dépenses, ils existent aussi des conséquences liées à la possibilité d'empilement de conteneurs telles que les mouvements improductifs (c'est-à-dire le fait de déplacer certains conteneurs pour accéder à d'autres) et le besoin de recourir parfois à des conteneurs vides pour éviter les problèmes de déséquilibre de navire.

Pour mieux rentabiliser l'utilisation de conteneurs, des tentatives d'amélioration sont continuellement menées dans les terminaux. Cela a soulevé des questions telles que la nécessité d'augmenter les capacités de stockage, l'instauration de moyens de réduction des retards, l'utilisation de techniques efficaces pour assurer la sécurité des ports et des biens. Des réponses satisfaisantes à ces questions s'imposent pour que les ports maritimes puissent faire face à l'accroissement du secteur industriel à l'échelle mondiale et à l'apparition de nouvelles puissances économiques. En effet, avec la mondialisation, les besoins d'importation et d'exportation de conteneurs ne cessent de croître. Cela a conduit à la création de très grands navires qui peuvent transporter des nombres très élevés de conteneurs. Ainsi, le nombre de conteneurs qui doivent être traités par unité de temps dans un port augmente aussi. Pour faire face à ce phénomène, deux possibilités de résolution existent :

- La première consiste à effectuer des travaux d'agrandissement du port, notamment de la zone de stockage. Les conséquences directes de cette option sont les perturbations de fonctionnement durant les périodes de travaux. En plus de cela, elle nécessite l'investissement d'une quantité d'argent non négligeable dont la rentabilité n'est pas garantie à cause des concurrences qui existent entre ports.
- La deuxième option consiste à trouver des moyens de gestion qui augmentent la productivité du port. Cette solution semble moins risquée que la précédente, mais elle nécessite des efforts intellectuels permanents.

Notre contribution porte sur cette dernière méthode d'amélioration. Ce choix est fait en se basant sur les avantages que comporte l'action de rendre plus performants les activités portuaires par rapport à celui de réaliser des travaux d'agrandissement de l'espace de stockage, car ce dernier aussi grand soit-il ne peut être productif s'il n'est

pas bien géré. À cela s'ajoute le fait que choisir la première option revient à réaliser des travaux d'agrandissement à chaque fois que la quantité de conteneurs gérés augmente, ce qui correspond à un risque de perturbations répétitives sur le fonctionnement du port et aussi à des dépenses d'argent dont la rentabilité n'est pas garantie.

L'accroissement continuel des nombres de conteneurs qui arrivent simultanément dans un port rend indispensable l'utilisation de moyens efficaces de gestion de l'espace de stockage. Il est évident que les méthodes de planification manuelles de stockage de conteneurs qui ne sont basées que sur les expériences des ouvriers et des cadres nécessitent beaucoup de temps et de labeur lorsque le nombre de conteneurs est très grand. Ainsi, elles doivent être remplacées par des logiciels ou bien des algorithmes efficaces et rapides.

Pour répondre à ces besoins, nous proposons dans cette thèse, des systèmes d'aide à la décision qui ont pour objectif de faciliter la gestion de quantités importantes de conteneurs en fournissant en des temps raisonnables de très bons plans de stockage. Deux cas sont distingués : le cas statique et le cas dynamique. Pour chacun d'eux, une étude analytique est d'abord menée afin d'étudier le contexte, d'identifier les contraintes physiques et opérationnelles, et de proposer un modèle mathématique de stockage de conteneurs. Après cela, des algorithmes de résolution pertinents sont proposés.

Organisation de la thèse

Ce travail est réparti en deux parties. La première concerne l'état de l'art, tandis que la deuxième porte sur nos approches de résolution. Chacune de ces parties est composée de chapitres dont les objectifs sont relatés ci-dessous.

Partie 1. État de l'art

Dans cette partie, nous présentons une étude bibliographique sur le problème de stockage de conteneurs. Elle comporte trois chapitres, qui sont décrits ci-dessous.

Chapitre 1. La conteneurisation

Ce chapitre relate l'histoire de la conteneurisation, ainsi que son impact dans la chaine logistique et dans la mondialisation.

Chapitre 2. Caractéristiques des ports et des terminaux à conteneurs

Ce chapitre décrit les différents types de terminaux à conteneurs portuaires ainsi que leurs rôles dans la chaine logistique.

Chapitre 3. Le problème de stockage de conteneurs

Dans le troisième chapitre, on décrit le problème de stockage de conteneurs proprement dit. Les différentes stratégies de stockage qui sont proposées dans la littérature y sont présentées, de même que des méthodes de résolution analytiques et des méthodes de ré-

solution numériques.

Partie 2. Approches proposées
La deuxième partie de cette thèse a pour objectif de présenter les différentes approches de résolution proposées pour la résolution du problème de stockage de conteneurs et donner des motifs prouvant leurs utilités et surtout leurs efficacités. Elle est composée de deux chapitres : le premier traite seulement le cas statique, alors que le deuxième concerne le cas dynamique.

Chapitre 4. Approches proposées pour la résolution du cas statique
Les objectifs du quatrième chapitre sont dans l'ordre : la présentation du contexte du problème étudié, la démonstration de la complexité du problème de stockage de conteneurs, la description des algorithmes employés pour résoudre le cas statique du problème considéré, ainsi que les résultats numériques prouvant l'efficacité de ces algorithmes.

Chapitre 5. Approches proposées pour la résolution du cas dynamique
Le cinquième chapitre est consacré à la présentation, la modélisation, l'étude théorique et la résolution du cas dynamique du problème de stockage de conteneurs.

Première partie

État de l'art

Chapitre 1

La conteneurisation

1.1 Introduction

La conteneurisation est le fait d'utiliser des conteneurs comme moyen de transfert de biens et de marchandises. Ce concept n'est apparu qu'au XX^{ème} siècle, mais depuis lors il est devenu un élément indispensable dans le domaine du transport. Plusieurs éléments ont contribué à son succès, parmi lesquels on peut citer son caractère multi-modal qui rend sa transition possible entre différents modes de transport. Cet atout, combiné à la possibilité de géo-localisation de cargo, grâce à des systèmes informatiques, a vite séduit les exportateurs et par la même occasion a contribué à la mondialisation du commerce. Devenant ainsi un outil international, le conteneur est par la suite standardisé grâce à des accords entre les compagnies de transport. Des améliorations et des spécifications ont été par la suite réalisées afin de rendre les conteneurs plus compatibles à certains types de cargaison. Toutefois, le conteneur n'est pas pour autant un élément infaillible, car même s'il a de nombreux avantages, il a rendu les contrôles manuels quasiment impossibles. Par conséquent, des efforts supplémentaires se sont imposés pour la création de moyens de contrôle technique.

Le reste de ce chapitre est organisé comme suit : les différents types de conteneurs sont présentés dans la section 1.2, l'historique de la conteneurisation ainsi que les changements engendrés sont exposés dans la section 1.3, les techniques d'inspection de conteneur sont relatées dans la section 1.4.

1.2 Description et propriétés du conteneur

Un conteneur est une caisse métallique rectangulaire qui sert à emmagasiner des éléments qui doivent être transportés d'un endroit à un autre. Grâce à la standardisation, les dimensions des conteneurs sont réglementées par la norme *ISO 668 : 1995*. L'unité de mesure de conteneur est l'*équivalent 20 pieds* (EVP), mais ils existent des conteneurs de 40 pieds (2 EVP), etc. Généralement, il y a plusieurs formes de conteneur, dont les plus connues sont : *standard*, *high-cube*, *hard-top*, *open-top*, *flatracks*, *plat, ventilated*, *insulated and refrigerated, bulk,* et *tank*. La Figure 1 est une illustration de ces différents types de conteneurs.

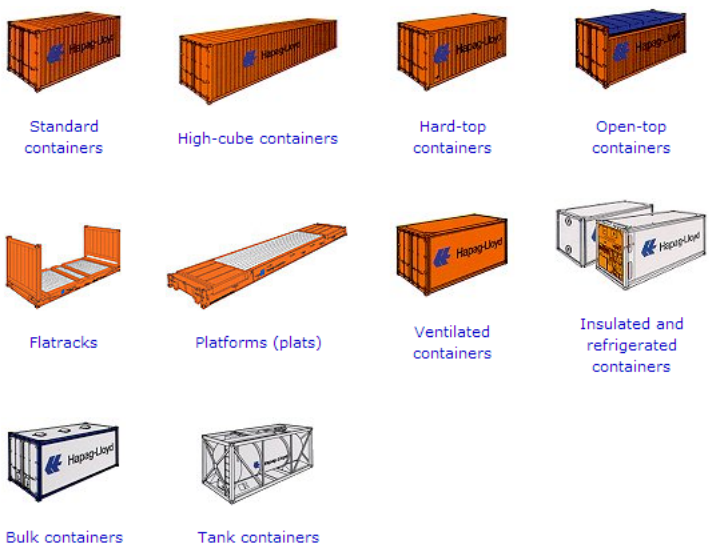


FIGURE 1 – Différentes formes de conteneurs

La forme standard

Les conteneurs standards sont connus comme étant des conteneurs à usage général. Ils peuvent être utilisés pour transporter tout élément sec, et sont fermés de tous les côtés. Néanmoins, une distinction peut être faite sur les types de conteneurs standards suivants :

- Conteneur standard avec des portes sur l'un ou sur les deux extrémités.
- Conteneur standard avec des portes sur l'un ou sur les deux extrémités et des portes sur toute la longueur de l'un ou des deux côtés.
- Conteneur standard avec des portes sur l'un ou sur les deux extrémités et des portes sur l'un ou sur les deux côtés.

Les dimensions d'un conteneur standard de 20 pieds (1 EVP) sont décrites dans le Tableau 1.1, alors que celles d'un conteneur standard de 40 pieds (2 EVP) sont mentionnées dans le Tableau 1.2. Ces informations peuvent être consultées dans [5].

Tableau 1.1 – Dimensions d'un conteneur standard 20 pieds

Dimensions internes			Portes ouvrantes		Poids			
Longueur [mm]	Largeur [mm]	Hauteur [mm]	largeur [mm]	hauteur [mm]	poids brut maximal [kg]	Poids à vide [kg]	Poids de la charge maximale [kg]	Volume [m^3]
5895	2350	2392	2340	2292	30480	2250	28230	33.2

Tableau 1.2 – Dimensions d'un conteneur standard 40 pieds

Dimensions internes			Portes ouvrantes		Poids			
Longueur [mm]	Largeur [mm]	Hauteur [mm]	largeur [mm]	hauteur [mm]	poids brut maximal [kg]	Poids à vide [kg]	Poids de la charge maximale [kg]	Volume [m^3]
12029	2350	2392	2340	2292	30480	3780	26700	67.7

Conteneur "high-cube"

Les conteneurs "high-cube" ont des structures similaires à celles des conteneurs standards, mais ils sont plus grands. Contrairement aux conteneurs standards, qui ont une hauteur de 2392 mm (8.6 pieds), les conteneurs "high-cube" sont hauts de 2697 mm (9.6 pieds). La plupart des conteneurs de ce genre sont des 40 pieds, mais ils existent aussi des 45 pieds de ce type. Les dimensions d'un conteneur "high-cube" 40 pieds sont notées dans le Tableau 1.3.

Une autre particularité de ce genre de conteneur est le fait qu'il possède des anneaux d'arrimage qui lui permettent de supporter des charges de plus de 1000 Kg.

Tableau 1.3 – Dimensions d'un conteneur "high-cube" 40 pieds

Dimensions internes			Portes ouvrantes		Poids			
Longueur [mm]	Largeur [mm]	Hauteur [mm]	largeur [mm]	hauteur [mm]	poids brut maximal [kg]	Poids à vide [kg]	Poids de la charge maximale [kg]	Volume [m^3]
12024	2350	2697	2340	2597	30480	4020	26460	76.3

Les conteneurs "high-cube" peuvent être utilisés pour tout type de marchandise sèche. Cependant, ils sont particulièrement adaptés au transport de marchandises légères et volumineuses.

Conteneur "hard-top"

Les parois des conteneurs "hard-top" sont généralement faits d'acier ondulé, alors que leurs planchers sont faits de bois. Ces conteneurs ont typiquement deux particularités structurelles distinctives. D'une part, ils sont équipés chacun d'un toit démontable en acier qui pèse environ 450 Kg. D'autre part, certains de ces toits présentent des points de fixation qui permettent aux chariots élévateurs de pouvoir les soulever. En outre, l'entête de la porte peut être pivoté vers l'extérieur, comme le montre la figure 2. Ces deux propriétés structurelles simplifient considérablement les processus de remplissage et de vidage du conteneur. En particulier, ces opérations peuvent être réalisées par une grue lorsque le toit est ouvert, ou bien par un chariot qui passe par la porte.

FIGURE 2 – Conteneur “hard-top”

Dans le cas du transport d’une cargaison qui a une grande hauteur, le toit du conteneur peut être laissé ouvert et fixé directement sur une parois à l’intérieur du conteneur.

Des anneaux d’arrimage, qui permettent de fixer la cargaison sont installés dans les rails latéraux supérieurs et inférieurs, et dans les poteaux d’angle et au milieu des parois latérales. Les anneaux d’arrimage qui sont sur les rails latéraux et sur les poteaux d’angle peuvent prendre des charges allant jusqu’à 2000 Kg, alors que ceux qui sont au milieu des parois latérales peuvent prendre des charges allant jusqu’à 500 Kg à condition que le toit soit fermé.

Généralement, les conteneurs “hard-top” sont de dimension 20 pieds ou bien 40 pieds, et sont généralement utilisés pour transporter des marchandises sèches.

Conteneur “open-top”

Les parois et les planchers des conteneurs “open-top” sont respectivement faits d’acier et de bois. Cependant, la particularité de ce genre de conteneur est le fait que le toit soit constitué d’arcs et d’une bâche amovibles. En outre, l’entête de la porte peut être pivoté vers l’extérieur. Il faut également noter que les arcs du toit ne permettent pas uniquement de fixer la bâche, mais ils contribuent aussi à la stabilité du conteneur. Des anneaux d’arrimage qui peuvent prendre jusqu’à 1000 Kg sont installés dans les rails latéraux supérieurs et inférieurs et dans les poteaux d’angle. Ce genre de conteneur convient à tout type de marchandise sèche.

Conteneur “flatracks”

Un conteneur “flatracks” est constitué d’une structure de plancher qui a une grande capacité de chargement et qui est composée des éléments suivants : une armature en acier, un plancher en bois tendre, et deux parois d’extrémité qui peuvent être soit fixes soit démontables. Les parois d’extrémité sont suffisamment stables pour permettre la fixation des moyens d’arrimage de la cargaison, ils permettent aussi la superposition de plusieurs conteneurs “flatracks” les uns au-dessus des autres. Un certain nombre d’anneaux d’arrimage, auxquels la cargaison peut être fixée, sont installés dans les rails latéraux, sur

les poteaux d'angle, et sur le sol. Ces anneaux d'arrimage peuvent prendre des charges qui ont des poids inférieurs ou égaux à 2000 Kg dans le cas d'un conteneur "flatracks" de 20 pieds, cette limite est fixée à 4000 Kg dans le cas d'un conteneur "flatracks" de 40 pieds.

Ce genre de conteneur est principalement utilisé pour transporter des cargaisons lourdes qui ont des hauteurs excessives et de grandes largeurs.

Conteneur plat

Un conteneur plat est constitué uniquement d'une structure de chaussée avec une capacité de chargement extrêmement élevée, il n'a pas de parois latérales ou d'extrémités. Cette grande capacité de chargement rend possible la concentration d'éléments lourds dans de petits espaces. Un conteneur plat est constitué d'une armature en acier et d'une structure de plancher en bois. Des anneaux d'arrimage, auxquels la cargaison peut être fixée, sont installés dans les rails latéraux. Ces derniers peuvent prendre des charges inférieures ou égales à 3000 Kg.

Les conteneurs plats sont généralement utilisés pour transporter des cargaisons surdimensionnées et très lourdes.

Conteneur ventilé

Les conteneurs ventilés ("ventilated") sont naturellement aérés et sont connus comme étant des conteneurs de café. L'aération est assurée par des ouvertures de ventilation dans les rails, comme le montre la figure 3. Ces ouvertures ne doivent pas être exposées à la pluie ou bien à l'humidité, pour éviter la dépréciation de la marchandise. Comme pour les autres conteneurs, les conteneurs ventilés sont également équipés d'anneaux d'arrimage, auxquels peut être fixée la cargaison. Ces anneaux peuvent stabiliser des charges inférieures ou égales à 1000 Kg, pour des conteneurs ventilés de 20 pieds.

FIGURE 3 – Ventilation d'un conteneur

Les conteneurs ventilés sont utilisés en particulier pour les marchandises qui ont besoin d'aérations. L'un des produits les plus connus de ce genre sont les grains de café vert, d'où le nom *"conteneur pour café"*.

Conteneur isolé et réfrigéré

Les conteneurs isolés et réfrigérés ("insulated and refrigerated") sont utilisés pour des produits qui doivent être transportés à une température constante au-dessus ou en-dessous de $0°C$. Ces produits sont divisés en produits réfrigérés et produits surgelés, en fonction de la température de transport spécifiée. Il s'agit principalement de fruits, de légumes, de viande, et de produits laitiers. Les conteneurs réfrigérés sont équipés de rails munis de crochets spéciaux qui sont fixés au plafond et qui permettent de suspendre de la viande. Cependant, une distinction peut être établie entre deux types de conteneurs isolés et réfrigérés : les conteneurs frigorifiques intégrés, et les conteneurs sabords ("porthole-container").

Conteneur frigorifique intégré

Ce type de conteneur réfrigéré a une unité de réfrigération intégrée qui permet de contrôler la température interne du conteneur. L'unité de réfrigération est disposée de manière à ce que les dimensions extérieures du conteneur correspondent au morne standard ISO et que le conteneur puisse être placé dans une cellule de navire porte-conteneur. Pendant le transport par bateau, les unités intégrées doivent être connectées au système d'alimentation électrique du bord. Le nombre de conteneurs frigorifiques pouvant être connectés dépend de la capacité du système d'alimentation électrique du navire. Pour le transport par voies routières ou ferroviaires, des groupes électrogènes sont utilisés dans la plupart des cas. L'air circule à travers le conteneur de bas en haut. En général, l'air chaud est soutiré de l'intérieur du conteneur, puis refroidi dans l'unité de réfrigération, ensuite soufflé dans le conteneur sous forme d'air froid. Pour assurer une bonne circulation de l'air froid, le plancher est muni de grilles. Des palettes séparent le plancher du conteneur et la cargaison, afin que l'air puisse circuler. En plus de cela, les parois latérales du conteneur sont ondulées, ce qui assure aussi un débit d'air satisfaisant. Dans la zone supérieure du conteneur, un espace suffisant (au moins 12 cm) doit également être prévu pour l'écoulement de l'air. A cet effet, lors du remplissage du conteneur, un espace libre adéquat doit être laissé au-dessus de la cargaison. La hauteur maximale de charge est indiquée sur les parois latérales des conteneurs. Pour assurer un flux d'air vertical de bas en haut, le remplissage du conteneur doit également être fait de manière appropriée et la cargaison doit être sensiblement rangée. En plus de la régularisation de la température, les unités intégrées permettent également un échange d'air frais contrôlé, par exemple l'élimination des produits métaboliques tels que le CO_2 et l'éthylène dans le cas du transport de fruits. Dans les unités de réfrigération, les quantités d'air fournies et aspirées sont mesurées et, selon le mode de fonctionnement, l'une de ces valeurs est utilisée pour contrôler l'air froid. La mesure de température peut être effectuée de diverses manières. Cependant, la température est affichée à l'extérieur de l'unité de réfrigération, de sorte que le fonctionnement de l'appareil puisse être contrôlé à tout moment.

Conteneur sabord

Ce type de conteneur n'est pas souvent appelé conteneur frigorifié, mais il est plutôt appelé conteneur isolé, puisqu'il n'a pas de dispositif frigorifique intégré. L'absence d'une unité de réfrigération permet à ce genre de conteneur d'avoir un volume interne plus grand

que celui d'un conteneur frigorifique intégré. À bord, l'intérieur de ce type de conteneur est alimenté en air froid par l'intermédiaire de l'installation de refroidissement centrale du navire. L'air s'écoule à travers le conteneur de la même manière que dans le cas du modèle intégré. L'air froid est soufflé par le bas, alors que l'air chaud est retiré au sommet. Hors du navire, la température est contrôlée par un système de réfrigération ou borne de "*clic-on units*".

Sur leurs parois d'extrémité opposées à leurs portes, ces conteneurs sont munis d'ouvertures pour les entrées et sorties d'air. En général, l'air est insufflé dans l'ouverture intérieure, puis distribué par l'intermédiaire des réseaux de diffraction dans le fond du conteneur, ensuite transporté vers le haut à travers la cargaison et évacué par l'ouverture de retour d'air. Ce type de conteneur nécessite également une circulation d'air adéquate. A cet effet, des conduits d'air appropriés doivent être prévus dans le plancher et au plafond, en plus de cela, le fret doit être judicieusement rangé.

Les conteneurs sabords n'ont pas d'afficheur de température intégré. Soit un tel affichage est installé dans les systèmes de réfrigération terminaux ("clic-on units"), soit les valeurs de la température peuvent être obtenues à partir des installations centrales de refroidissement du navire.

Si les conteneurs sabords sont dotés de "clic-on units" quand ils sont à terre, alors ils ne répondent plus aux normes de dimensionnement ISO.

Les portes constituent des points faibles aussi bien pour les conteneurs frigorifiés de type intégré que ceux de type sabord. Le fait de mettre des joints aux portes en caoutchouc, ou bien d'effectuer une mauvaise manipulation peut endommager les portes à tel point qu'elles ne se ferment plus correctement et qu'elles ne soient plus en mesure d'empêcher à l'eau de pluie de s'infiltrer dans le conteneur. Pendant le transport de marchandises réfrigérées ou congelées, l'infiltration d'eau peut conduire à la détérioration des marchandises ou à la formation de glace dans la zone de la porte. En outre, la capacité de réfrigération doit être augmentée pour compenser les pertes dues aux fuites d'air froid.

Conteneur vrac

Un conteneur vrac ("bulk") a trois trappes de chargement dans le toit, chacune d'elles ayant un diamètre qui est approximativement égal à 455 mm. Il dispose également de deux trappes d'évacuation sur le côté de la porte. Ces dernières sont parfois équipées de tubes de déchargement courts qui servent à guider la cargaison en vrac.

De tels conteneurs peuvent également être utilisés pour le fret en général. Cependant, des anneaux d'arrimage doivent être montés dans les rails latéraux supérieurs pour permettre la fixation de la cargaison. Toutefois, les conteneurs vrac sont généralement utilisés pour le transport de marchandise en vrac, tels que les céréales, les aliments pour animaux, et les épices.

Conteneur citerne

Un conteneur citerne ("tank") doit être rempli d'au moins de 80%, pour éviter des déferlements dangereux du liquide. D'autre part, il ne doit pas être rempli de plus de 95%, car dans le cas contraire, il n'y aurait pas suffisamment d'espace libre pour l'expansion

thermique. L'étendue de la dilatation thermique peut être calculée pour chaque cargaison avec les formules suivantes :

$$\Delta V = V_a \cdot Y \cdot \Delta T$$

$$V_e = V_a(1 + Y \cdot \Delta T)$$

ΔV : l'écart entre le volume initial et le volume après la dilatation
V_a : le volume initial, à la température a
V_e : le volume final, à la température e
Y : le coefficient de dilatation (thermique)
ΔT : la différence de température en kelvin

Les conteneurs citernes sont utilisés pour des cargaisons liquides, telles que : des denrées alimentaires (jus de fruits, huiles douces, etc.), et des produits chimiques (pétrole, gazole, etc.).

Si la cargaison nécessite un transport à température contrôlée, les conteneurs citernes peuvent être munis d'une isolation ou de chauffages. La température de la cargaison peut être contrôlée avec précision en utilisant des sondes de température.

1.3 Histoire de la conteneurisation

Le conteneur inter-modal est une boite en acier dont les dimensions sont standardisées afin de pouvoir l'utiliser pour transporter des marchandises par bateau, camion, train, et avion. Il s'agit d'une technologie dont l'utilisation a un impact social et économique. Cet impact a été initialement estimé par les entrepreneurs de l'industrie maritime, notamment les autorités portuaires et les compagnies maritimes. Mais, comme pour de nombreuses technologies extrêmement efficaces, l'impact du conteneur s'est élargi et affecte le développement mondial.

1.3.1 La nécessité du conteneur

Dans les années 1950, la plupart des marchandises transportées par bateau sur de longues distances était expédiée en vrac, emballées dans des boîtes, des sacs, des barils, ou autres récipients relativement petits dont les natures et les tailles varient selon les types de bien contenus. Un des inconvénients de ce système est le coût important en main d'œuvre et en temps qu'il nécessite pour effectuer les chargements et les déchargements des navires d'une façon qui minimise les dommages aux biens. Des analyses publiées dans ([73] : pages 21, 33-34) ont révélé que 60% des coûts de transport d'une cargaison maritime sont constitués de coûts bâbord, et que la manutention du fret représentait environ 37% du coût total. Ces coûts comprennent non seulement la main d'œuvre, mais aussi les pertes de temps et les dommages (y compris le vol) subits par les cargaisons. Selon Cudahy [21], le temps de chargement et de déchargement d'un cargo pourrait avoisiner son temps de voyage en mer.

L'exception était le transport d'un seul type de bien, comme le pétrole. Pour ce genre de marchandise, des navires et des installations portuaires ont été spécialisés pour permettre des chargements et des déchargements plus rapides et à moindre coût. Ce transport de vrac spécialisé était devenu industrialisé, contrairement au transport de marchandises diversifiées.

Les coûts élevés du transport maritime ralentissaient le commerce international. Ainsi, quelques tentatives ont été faites pour remédier à cela. Par exemple, l'armée américaine avait commencé à utiliser des conteneurs en métal de dimension 2.63m×1.93m×1.86m pendant la seconde guerre mondiale et a continué à le faire dans les années 1950, [4]. Par la suite, des tentatives commerciales, qui ont eu une incidence beaucoup plus grande, ont été faites par des entreprises aux États-Unis d'Amérique, en particulier celles qui étaient dirigées par un ancien magnat du transport maritime, Malcom MacLean. L'idée de ce dernier consistait à utiliser des conteneurs similaires à ceux utilisés par l'armée américaine mais qui ont des tailles plus grandes et qui peuvent être transportés par camion ou par train (donc inter-modal). De ce fait, le processus de chargement de marchandises sur des navires serait décomposé en deux étapes. Dans un premier temps, les marchandises sont mises dans des conteneurs à l'extérieur du port (éventuellement dans des endroits qui sont situés à proximité des lieux de fabrication ou d'assemblage de marchandises). Ensuite, une fois arrivés au port, les conteneurs sont chargés sur des navires. Le processus de déchargement est presque similaire, sauf que les retraits de marchandises transportées dans les conteneurs se font dans des points de distribution ou de vente situés à l'extérieur du port. Les entreprises de Malcom et une autre entreprise nommée *Maston Navigation Compagnie*, utilisaient cette technologie avec succès le long d'un certain nombre de routes maritimes entre la fin des années 1950 et le début des années 1960 ([73] : pages 54-68). C'est le début de la révolution du conteneur.

1.3.2 Normalisation et vulgarisation

Le conteneur inter-modal est devenu le récipient privilégié pour transporter la plupart du fret maritime depuis les années 1960 pour deux raisons. L'une est le succès des entreprises qui l'ont initié, telle que celle de Malcom (qui fut rebaptisée "*Sea-Land Service*", soulignant le caractère inter-modal de son activité). Le service qu'a rendu "*Sea-Land Service*" à l'armée américaine durant la guerre du Vietnam a prouvé l'efficacité de cette entreprise et a contribué à sa croissance ([73] : pages 176-188). L'autre raison est la normalisation des tailles de conteneur à travers l'industrie du transport, ce qui a permis d'importants investissements en navires et en équipements de manutention de conteneur. Durant les premières années d'utilisation, différentes sociétés ont utilisé des conteneurs qui sont adaptés à leurs industries ou bien à des circonstances particulières, telles que des facteurs affectant la taille du conteneur, y compris les navires qu'ils possèdent, le type de marchandise transportée, des limites légales de longueur ou de poids des charges transportées sur des routes ou bien dans les marchés qu'ils desservent, et des limites similaires pour des voyages ferroviaires. Toutefois, des accords internationaux ont été rapidement trouvés au début des années 1960 sur l'ensemble du secteur industriel concernant les dimensions des conteneurs. Des accords ont été également trouvés sur la robustesse des conteneurs, afin de rendre possible leurs empilements et leurs transports par bateau, camion, train, et avion. Les extrémités des conteneurs doivent être suffisamment solides pour pouvoir résister aux chocs qui sont produits lorsque des wagons se heurtent, ces chocs sont plus intenses que ceux qui se produisent sur des navires ou bien sur des camions. Des normes ont été également fixées pour les appareils de manutention de conteneur ([73] : pages 127-149 ; [14] : pages 12-16). Les compromis réalisés à cette époque ont donné naissance aux tailles de conteneur les plus répandues de nos jours.

1.3.3 Impact du conteneur sur le travail portuaire

L'impact de l'utilisation accrue de conteneurs a été immédiatement ressenti par les travailleurs portuaires, avec les économies réalisées en gain de temps dans les opérations de chargement et de déchargement qui diminuaient ainsi le nombre de travailleurs nécessaires. Des études ont montré que la quantité de marchandises qui peuvent être chargées ou bien déchargées par travailleur a augmenté grâce à l'utilisation de conteneurs ([14] : pages 235-236). Ces changements ont naturellement rencontré des craintes de la part des travailleurs et des syndicats, résultant à de grands conflits entre les dockers et les compagnies maritimes durant les années 1980. Le résultat final était une énorme baisse des nombres de dockers, on peut citer l'exemple de la côte ouest des États-Unis où une baisse de plus de deux-tiers a été enregistrée entre 1952 et 1972. Cette baisse a également touché le Royaume-Uni, dont le nombre de dockers est passé de 70000 à moins de 10000, entre le début des années 1960 et la fin des années 1980 ([14] : pages 237-238). Ces changements ont eu lieu en dépit de la croissance de plus de 600% de l'exportation mondiale entre les années 1950 et 1973 ([16] : page 63).

1.3.4 Impact du conteneur sur d'autres pratiques en matière de technologie et d'affaires

La nature du travail à quai a évidement changé puisque la manipulation des conteneurs exige des compétences plus techniques pour l'utilisation des machines. La normalisation de la taille des conteneurs et des outils de manutention signifie que le même équipement de manutention de cargaison pourrait être utilisé pour une grande variété de marchandise. De plus, les navires pourraient être conçus dès le départ pour transporter des conteneurs. L'incertitude dans le transport maritime a été également réduite, car il est plus facile pour une entreprise d'expédition de calculer la vitesse de chargement ou de déchargement des conteneurs que celle de petits colis.

Ces progrès ont favorisé l'accroissement des investissements dans des navires et des compagnies maritimes depuis les années 1970 ([14] : pages 72-76), et aussi la création de navires de plus en plus spacieux. De plus, l'augmentation de l'ampleur des opérations des compagnies maritimes a favorisé l'intégration de transports terrestres. Un exemple clé est la compagnie "*Sea-Land*". Elle était au début une entreprise de camionnage, et par la suite, elle s'est fusionnée avec la compagnie ferroviaire CSX en 1986 ([22] : pages 160-166).

1.3.5 Impact sur les ports et les villes

Les progrès dans les coûts abaissés de main d'œuvre, de même que l'accélération des processus de chargement et de déchargement, ainsi que l'augmentation de la taille des navires se sont produits en parallèle avec des changements dans les ports eux-même. L'apparition de grands navires rimait avec la nécessité d'eaux plus profondes. Mais plus important encore, est le fait que le commerce conteneurisé nécessite plus d'espace. Les conteneurs étant eux même des endroits de stockage, il n'était plus nécessaire d'avoir des entrepôts de marchandise dans les ports. Par contre, il est devenu primordial d'avoir des espaces où stocker les conteneurs eux-même et le volume de marchandise supplémentaire

occasionné par la baisse des coûts de transport. Dans de nombreux pays, ce phénomène a causé le déplacement des activités portuaires qui étaient proches des centres villes vers des endroits moins développés. Des exemples de ce constat sont le développement de Tilbury, qui est devenu le principal port de Londres ([18] : pages 132-144), et le déplacement des opérations de fret qui se déroulaient sur les quais de New York vers d'autres localités à Elizabeth et dans le New Jersey. Dans certains cas, la croissance des ports a non seulement pris la forme d'un accroissement direct, mais aussi de la consolidation de plusieurs aménagements dans des villes voisines ([80] : pages 129-131).

Dans de nombreux cas, l'ampleur du transport de conteneurs a mis en évidence la valeur de la coopération régionale. En Californie par exemple, la concurrence entre les ports de Los Angeles et de Long Beach pour le trafic de conteneurs a fait place à une plus grande coordination entre eux dans les années 1980 ([41] : pages 88-93). Dans la grande région de New York, l'autorité portuaire de New York et celle du New Jersey ont joué un rôle clé dans cette coordination régionale.

La conteneurisation a contribué à des changements dans la localisation de l'industrie et du travail dans les régions. L'avantage de fabriquer dans des zones portuaires des éléments destinés à l'exportation disparaît puisqu'il existe désormais un transport multi-modal à bas prix. Au lieu de cela, la fabrication pourrait se propager dans des aménagements conçus pour faciliter la circulation des camions transportant des conteneurs, plutôt que de fabriquer des produits (à exporter) le long des quais. Ces changements ont été spectaculaires à New York, par exemple, avec l'accroissement vertigineux de la fabrication dans la ville alors que l'expédition de conteneurs vers les régions intérieures a été en plein essor dans le New Jersey ([97] : pages 48-55 ; [73] : pages 98-100).

1.3.6 Impact global du conteneur

L'impact le plus profond du conteneur est fait sur l'économie mondiale dans son ensemble. Dans le monde entier, au début des années 2000, 300 millions de conteneurs de 20 pieds ont été transportés par voie maritime chaque année, avec plus d'un quart de ces expéditions en provenance de Chine ([73] : pages 277). Comme le dit Slack dans [90] : *"La mondialisation et la conteneurisation entretiennent une relation réciproque. Il y a peu de doute que l'expansion du commerce mondial et l'expansion des systèmes de fabrication mondiale auraient été impossible sans les efficacités et les économies que la conteneurisation a apporté."*

La mondialisation est, à juste titre, l'objet de nombreux débats. Nous avons vu comment les conteneurs ont réduit le nombre d'employés dans les ports individuels. Au delà, la mondialisation a entraîné le déplacement de l'emploi entre villes, régions et pays. Il a également réduit les coûts pour les consommateurs et a rendu possible la livraison d'une variété beaucoup plus large de produits dans de nombreux marchés. La mondialisation a affecté non seulement les économies, mais aussi l'environnement, la politique et la culture. Le conteneur, une simple technologie qui est destinée à accélérer les chargements et les déchargements des marchandises, a joué un rôle important dans ces changements.

1.4 La conteneurisation et la sécurité mondiale

Les attaques terroristes perpétuées sur la ville de New York, et la destruction du World Trade Center le 11 Septembre 2011 ont éveillé la conscience collective sur la vulnérabilité de tous les modes de transport par rapport à des attaques terroristes. Selon Fritelli [42], la vulnérabilité de la chaîne logistique maritime de conteneurs face à d'éventuels actes terroristes a conduit à une utilisation accrue de plusieurs technologies d'inspection de conteneur. L'intégrité d'un conteneur lors de son transport d'un point quelconque A à un autre point B dans la chaîne logistique ne peut pas être assurée. En effet, selon Van De Voort et al. [107], il suffit de disposer d'assez de temps, d'opportunités, et d'un emplacement discret, pour que des malfaiteurs soient en mesure d'ouvrir un conteneur et d'altérer son contenu.

Le terme "sécurité portuaire" est une abréviation du vaste effort fourni pour garantir l'ensemble de la chaîne d'approvisionnement maritime, depuis la porte de l'usine de fabrication jusqu'à la destination finale des produits. La sécurité des conteneurs n'affecte pas uniquement celle des ports, mais elle concerne aussi n'importe quelle localité sur terre. D'ailleurs, selon Cohen [29], un conteneur est un excellent moyen de transport d'armes de destruction massive car il est indispensable et omniprésent. Il n'est pas possible d'inspecter manuellement et de façon complète tous les conteneurs qui entrent dans un pays. Ce qui rend indispensable l'utilisation et le développement de technologies de contrôle et de suivi. La section suivante porte sur les technologies qui sont actuellement utilisées pour appuyer l'inspection de conteneurs afin de détecter des matériaux qui pourraient être utilisés pour fabriquer des armes atomiques de destruction massive.

1.4.1 Technologies actuelles pour inspecter des conteneurs fermés

Les méthodes d'inspection de conteneurs les plus connues sont : la méthode non-intrusive (l'imagerie avec les rayons-x et les rayons-gamma) et la technique de détection de rayonnement.

L'utilisation de rayons-x et de rayons-gamma

La technologie d'inspection non intrusive (NII) offre la possibilité de voir le contenu des conteneurs sans avoir à effectuer les processus de dépotage et d'inspection physique qui nécessitent énormément de temps [31]. Cependant, cette technologie nécessite une importante source de radiation (une source de photon) et un moyen de détecter les indices caractéristiques d'éléments suspects. Selon Jones [54], cette double nécessité conduit au grand besoin de faisceaux d'énergie en photon et des détecteurs d'image performants. Le système SAIC VACIS et le Rapidscan EagleTM système satisfont ces exigences.

L'utilisation du système de contrôle NII requiert des moyens de gestion de quantités importantes d'informations d'une façon efficace qui ne ralentie pas le trafic. A cet effet, est conçu le SAIC VACIS qui est un système intégré d'inspection de conteneur capable d'intégrer et de gérer des images et des données (voir Figure 4). Lorsque ces propriétés sont combinées à des scanners qui sont très rapides, la compagnie affirme qu'une seule voie peut gérer plus de 300 conteneurs par heure [89].

FIGURE 4 – Système d'inspection intégré de conteneur SAIC

Contrairement au système SAIC VACIS, le système Rapidscan Eagle est un dispositif mobile d'inspection de cargo qui fournit en temps réel des images radiographiques de plusieurs objets inspectés y compris des conteneurs, comme le montre la figure 5.

FIGURE 5 – Rapidscan Eagle

Le Rapidscan Eagle qui est conçu pour répondre au besoin d'imagerie élevé de rayons-x pour l'inspection de fret, emploie un générateur de rayons-x à accélération linéaire de 9 méga volt et délivre 425 millimètres de pénétration en acier. En conséquence, ce système peut inspecter une large gamme de marchandises, y compris les camions et les conteneurs densément chargés, éliminant ainsi la nécessité d'inspections manuelles lentes et coûteuses. Les camions sont automatiquement déplacés à travers un faisceau de rayons-x stationnaire sur un système de mouvement de cargaison. Selon la compagnie, tout un conteneur de 20 pieds peut être consulté en moins de 30 secondes [96].

Détection de radiation

Ce système d'inspection emploie des grues spécifiques, appelées VeriSpreader, qui sont dotées de moyens d'identification et de détection de radiation (voir Figure 6). Ainsi, lors des chargements et des déchargements de navires, une technologie de scan passive et des algorithmes d'identification sont employés pour détecter des sources de rayons-gamma et de neutrons [32]. Ces grues sont conçues de sorte que leurs épandeurs puissent incorporer des détecteurs de neutrons sensibles et de rayons-gamma. Il faut noter que l'épandeur d'une grue est la partie qui entre en contact direct avec les conteneurs lors des chargements et des déchargements de navires.

FIGURE 6 – VeriSpreaderTM

En plus des technologies de sécurité présentées dans ce chapitre, ils existent diverses innovations qui sont continuellement développées dans des laboratoires afin d'améliorer les techniques de contrôle de conteneurs [93].

1.5 Conclusion

Dans ce chapitre, nous avons expliqué ce que c'est un conteneur, et exposé les différentes variétés de conteneur qui existent ainsi que leurs utilités. L'histoire détaillée de la

conteneurisation est présentée de même que les changements qu'a apporté cette technologie dans le monde entier. Certes, la conteneurisation a facilité le transport de marchandises et les exportations mondiales en diminuant considérablement les coûts de transport et les temps de chargement et de déchargement des navires ; mais elle a aussi rendu quasiment impossibles les contrôles manuels de cargaisons, rendant ainsi indispensable l'utilisation de moyens de contrôle techniques.

Les changements imposés par l'utilisation de conteneurs ont aussi affecté les modes de fonctionnement des ports. Ces derniers sont désormais confrontés à la nécessité de gérer d'énormes quantités de conteneurs en plus des équipements de manutention de conteneurs. Ceci a rendu indispensable la création de services de gestion de conteneurs, appelés *terminaux à conteneurs*, dans chaque port. Une description plus détaillée de ces terminaux à conteneurs ainsi que les activités qui y sont réalisées sont présentées dans le chapitre suivant.

Chapitre 2

Description des ports et des terminaux à conteneurs

2.1 Introduction

Dans ce chapitre, nous expliquons ce que c'est un port et les rôles qu'il peut jouer dans la chaîne logistique. L'intégration de ce dernier a été faite de façon progressive. De nos jours, des améliorations et des modernisations sont visibles sur les structures et les modes de fonctionnement des ports. La conteneurisation a été un puissant catalyseur dans ce processus de développement. D'ailleurs, jusqu'à présent, cet outil, qu'est le conteneur, reste impliqué dans les travaux de recherche concernant les développements des ports.

Certains ports possèdent même plusieurs terminaux à conteneurs. Nous allons également voir, dans ce chapitre, les composantes ainsi que les rôles d'un terminal à conteneurs. Ce dernier a aussi bénéficié des effets de la modernisation, surtout sur le plan technique. Les principales différences entre les terminaux à conteneurs résident au niveau de leurs équipements, qui à leurs tours sont continuellement perfectionnés afin d'augmenter leur efficacité et leur rendement. Des descriptions plus détaillées sur les matériels de travail utilisés dans les terminaux à conteneurs sont exposées dans les sections qui suivent.

Le reste de ce chapitre est organisé comme suit : dans la section 2.2 sont données des définitions d'un port et des explications sur leurs utilités, la section 2.3 porte sur les structures et les développements des ports, les descriptions et les rôles des terminaux à conteneurs sont exposés dans la section 2.4, une conclusion est donnée dans la section 2.5.

2.2 Définitions et rôles des ports

Les ports jouent des rôles multiples dans l'industrie maritime et font partie d'un réseau complexe. Ce sont des interfaces reliant la mer et le transport terrestre. Cependant, il existe un grand nombre de définitions de port. Par conséquent, nous allons en citer quelques unes, en commençant par celle donnée par Stopfords ([94] : page 81) qui caractérise un port de :

"Une zone géographique où les navires sont mis à côté de la terre afin d'y charger et décharger des marchandises - habituellement une zone d'eau profonde comme une baie ou l'embouchure d'un fleuve".

Cette définition est assez simple, mais elle donne un aperçu sur le rôle fondamental d'un port. En même temps, il est important d'y comprendre que le rôle d'un port est plus complexe que juste le fait d'être un emplacement en bord de mer. Aujourd'hui, les ports sont des éléments clés dans le système du transport mondial, car sans leur existence les navires seraient inutiles dans le transport de marchandises, puisqu'ils n'y auraient aucun endroit où ils pourraient être chargés ou déchargés. Par conséquent, les ports peuvent être vus comme des facilitateurs du commerce multi-modal (entre terre et mer).

L'utilité de l'existence de ports dans le monde se fait ressentir même dans les régions qui n'ont pas de côtes maritimes, car ces dernières bénéficient du commerce maritime des autres régions. C'est le cas des pays enclavés tels que la Suisse, la République Tchèque, l'Autriche, la Hongrie, la Slovaquie et-cetera, qui bénéficient du marché européen. Cependant, il faut noter que certains de ces pays ont de petits ports fluviaux qui sont reliés à des océans par des cours d'eau navigables.

Un autre élément qui corrobore l'utilité des ports dans le monde est le fait que l'ampleur du commerce maritime a plus que doublé de volume depuis les années 1980. D'ailleurs 90% du commerce mondial a été traité par des ports, selon Lee et Hsu [74]. Les ports sont des catalyseurs de l'activité économique d'une région, ce qui motive la création de grands ports dans les régions qui ont les opportunités. Dans [76], Fynes et al. stipulent que les ports sont des éléments clés dans la détermination de la compétitivité globale des régions et des pays au niveau économique. Les activités portuaires sont, dans certains pays, le principal conducteur économique, on peut citer l'exemple de Singapour. Cela signifie que les ports sont plus que de simples zones géographiques où sont effectués des chargements et des déchargements de cargaisons, et qu'ils ont des rôles plus complexes. Les études récentes de Hall et al. ([52] : page 83) décrivent le rôle des ports dans une chaîne logistique globale comme étant :

"Une manifestation physique des fonctions logistiques que ces endroits servent dans le commerce mondial et global sur les matières premières."

Cette définition donne un aperçu sur la complexité de la chaîne d'activités qui sont en rapport avec les ports. Cela se reflète dans la définition de port présentée dans le rapport annuel du port d'Anvers qui stipule que :

"Le port est comme un lien dans une chaîne logistique interconnectée qui s'étend de l'avant pays d'outre mer à l'arrière-pays continental, dans un flux continu de marchandises sans frontière."

Cette vision du port est aussi consentie par Notteboom qui affirme dans [82] que :

"Les ports européens fonctionnent de plus en plus non pas comme des entités individuelles qui gèrent des navires, mais plutôt comme des chaînes d'approvisionnement et des réseaux."

Selon Lam et Yap [70], les performances d'un port procurent des avantages compétitifs à sa région. Ces derniers affirment également que les ports ont des rôles de catalyseurs économiques pour les régions, et les définissent comme étant :

"Des plate-formes intégrées servant de base pour la production, le commerce, la logistique, et le transfert d'informations."

Après ces différents constats, on peut dire que les régions ont intérêt à encourager les commerces maritimes, et que les ports doivent être en mesure de répondre aux exigences de la clientèle afin d'être attractifs. L'intensité des opérations qui peuvent être réalisées dans un port dépend de la taille du port et de ses infrastructures, car certaines cargaisons ont des besoins spécifiques (c'est le cas des produits surgelés).

Les avantages économiques que comportent les activités portuaires n'ont pas manqué de créer des compétitions. Trois niveaux de compétition sont définis dans [46] :

1. Compétition entre gammes de port.
2. Concurrence entre ports d'une même gamme.
3. Concurrence entre opérateurs dans un même port.

Selon Huybrechts et al. [51], la compétition entre ports est influencée par cinq principaux éléments, qui sont :

1. Les demandes spécifiques des consommateurs.
2. Les facteurs spécifiques de production.
3. Les industries qui collaborent avec les opérateurs.
4. Les compétences spécifiques à chaque opérateur et à ses rivaux.
5. Les structures des autorités portuaires et d'autres organismes publics.

Ces cinq points portent sur la balance commerciale entre l'importation et l'exportation, l'intégration entre les industries et les opérateurs, les niveaux de compétence des opérateurs et de leurs rivaux, et les structures des autorités portuaires.

2.3 Structures et développements des ports

Comme mentionné ci-dessus, les ports doivent adapter leurs infrastructures aux normes des cargaisons afin de satisfaire leurs clientèles. En plus de cela, les autorités portuaires ainsi que les armateurs doivent changer de tactiques en fonction des fluctuations de marchés et des nouvelles opportunités de marchés qui apparaissent. D'ailleurs, comme le dit Branch dans ([6] : page 396), les développements des ports sont dirigés par la recherche de marchés. Il faut aussi noter que le niveau d'infrastructure d'un port détermine son champ d'opération, car chaque type de cargaison a ses propres exigences qui englobent aussi bien les besoins sur les quais (chargement et déchargement) que les nécessités d'emplacements de stockage et de moyens de transport. Cependant, les investissements en infrastructures portuaires sont très coûteux, mais cela n'empêche qu'ils valent la peine d'être réalisés, car comme le dit aussi Branch ([6] : page 396), ils sont cruciaux pour le maintient des avantages compétitifs d'un port.

L'importance socio-économique des ports a motivé des organismes socio-économiques à investir dans ce domaine. Ces derniers louent des infrastructures et des zones portuaires à des entreprises logistiques. Dans [6], Branch décrit un processus de privatisation de ports, dans lequel, des gouvernements externalisent la gestion de certains ports afin de leurs rendre plus attractifs. Les objectifs de ce processus sont : l'augmentation des investissements de capitaux étrangers, l'accroissement de la productivité, et la stimulation du commerce. En effet, la modernisation des ports est un élément clé dans le développement du commerce régional et des centres de distribution. Cependant, malgré ces efforts continuels de modernisation, tous les ports ne sont pas au même niveau de développement. Dans [94], quatre niveaux de développement, qui sont basés sur les niveaux d'infrastructure, sont décrits. Ils sont constitués de :

Niveau 1 : Petits ports locaux.
Niveau 2 : Grands ports locaux.
Niveau 3 : Grands ports régionaux.
Niveau 4 : Centres de distribution régionaux.

Les petits ports locaux (niveau 1) ont chacun un terminal à usage général avec un quai et des grues pour d'éventuelles opérations de stockage. Ces ports reçoivent et expédient de petites quantités de fret pour le transport local (intra-régional dans la plupart des cas). Ils sont essentiellement desservis par des navires maritimes à courtes distances, qui peuvent transporter différents types de cargaisons tels que des conteneurs, des palettes, et des produits emballés. Ces types de port se trouvent principalement dans les pays en développement et dans les zones rurales des pays développés.

Les grands ports locaux (niveau 2) sont plus développés que les petits ports locaux. Ils peuvent accueillir une grande variété de marchandises, et possèdent des terminaux à usage général. Leurs infrastructures sont plus personnalisées pour de grandes exploitations. Ces ports ont souvent des terminaux qui peuvent accepter et amarrer de grands vraquiers (navires destinés au transport de marchandises solides en vrac).

Les grands ports régionaux (niveau 3) ont la possibilité de gérer de très grandes cargaisons qui ne peuvent pas être prises en charge par des ports de niveau 1 ou 2. Ils ont des équipements spécialisés qui leurs permettent d'effectuer des opérations plus importantes que celles qui peuvent être faites dans des grands ports locaux (niveau 2). En plus de cela, ils ont généralement de grandes capacités de stockage, plusieurs terminaux, plus de matériels de manutention, et sont souvent connectés à de vastes réseaux de transport (routes pour camion, chemins de fer, et-cetera).

Les centres de distribution régionaux (niveau 4), servent de plate-formes dans leurs régions et permettent la distribution de cargaisons aux niveaux intra-régional et inter-régional. Ces ports opèrent dans des marchés spécifiques (exemple le marché européen). Ils reçoivent des marchandises en provenance d'autres horizons, ensuite, ils les redistribuent vers diverses destinations par différents modes de transport (maritime, fluvial, routier, ferroviaire, ou bien par des tuyaux). Selon Manga ([77]), ces cinq modes de distribution sont les principaux.

Ces genres de ports ont des terminaux spécialisés à différents types de cargaisons, ils

disposent également d'équipements de manutention sophistiqués et peuvent amarrer les plus grands navires qui existent. Ils ont de vastes réseaux de transport pour le transbordement de cargaisons. Parmi eux, ont peut citer le port de Rotterdam, qui est le plus grand centre de distribution de l'Europe, suivi du port d'Anvers [94].

Dans [52], Hall et al. décrivent les centres de distribution comme des ports essentiels, par où passent les lignes maritimes des riches et vastes marchés. Ces ports demeurent attractifs grâce à leurs flux de marchandises, même si certains d'entre eux ne se trouvent pas dans les lieux très accessibles. C'est le cas des ports d'Anvers et de Hambourg, qui sont à côté d'embouchures et qui nécessitent plus de temps de voyage, comparés aux ports du Havre et de Rotterdam qui sont situés près d'océans.

L'intégration des ports dans la chaîne logistique globale s'est faite de façon progressive, selon Pettit et Beresford [88]. Ces derniers affirment que l'effet de la mondialisation a forcé les ports à s'adapter afin de maintenir leurs avantages de compétitivité. Ainsi, en s'intégrant dans la chaîne logistique, ils fonctionnent pour maintenir leurs compétitivités, tandis que l'objectif global de cette dernière est d'améliorer la productivité.

Après ces observations, on peut affirmer que le rôle des ports s'est amélioré au fil du temps, et qu'ils occupent aujourd'hui une place capitale dans le commerce international.

Le Tableau 2.1 est un classement des vingt ports les plus grands du monde suivant les flux de conteneurs, d'après [108].

Tableau 2.1: Les 20 premiers ports conteneurisés du monde en 2009

Rang	Port	Pays	Volume (mille EVPs)
1	Singapour	Singapour	25.87
2	Shanghai	Chine	25.00
3	Hong Kong	Chine	21.04
4	Shenzhen	Chine	18.25
5	Busan	Corée du Sud	11.98
6	Guangzhou	Chine	11.19
7	Dubaï	Emirates arabe	11.12
8	Ningbo-Zhoushan	Chine	10.50
9	Quindao	Chine	10.26

10	Rotterdam	Pays-Bas	9.74
11	Tianjin	Chine	8.70
12	Kaoshsiung	Taïwan, Chine	8.58
13	Port Klang	Malaisie	7.31
14	Anvers	Belgique	7.31
15	hamburg	Allemagne	7.01
16	Los Angeles	États Unis d'Amérique	6.75
17	Tanjung Pelepas	Malaisie	6.00
18	Long Beach	États Unis d'Amérique	5.07
19	Xiamen	Chine	4.68
20	Leam Chabang	Taïland	4.62

Le tableau ci-dessus reflète le développement de l'économie asiatique (surtout chinoise) et l'amélioration de l'industrie portuaire dans cette région. Selon [108], les choses étaient différentes vingt ans auparavant lorsque la moitié des 20 premiers ports mondiaux était en Amérique et en Europe.

2.4 Description et rôles des terminaux à conteneurs

D'une façon générale, un terminal à conteneurs est une installation où les conteneurs sont transbordés entre différents moyens de transports.

Le transbordement peut être fait entre navires et véhicules terrestres (camions ou trains), dans ce cas, le terminal est décrit comme étant un *terminal maritime*. Alternativement, le transbordement peut être fait entre deux véhicules terrestres, généralement entre trains et camions, dans ce cas le terminal est décrit comme étant un *terminal intérieur*.

Les terminaux maritimes se trouvent généralement dans de grands ports, alors que les terminaux intérieurs ont tendance à être situés dans de grandes villes, avec de bonnes liaisons ferroviaires vers les terminaux à conteneurs maritimes.

Les principales fonctions d'un terminal à conteneurs maritime sont : la réception, le stockage, la préparation, et le chargement de conteneurs entrants (qui arrivent au port par voie maritime, et qui quittent par voie terrestre) ou sortants (qui arrivent au port par voie terrestre et quittent par bateau).

- La réception de chaque conteneur qui arrive au port englobe : son déchargement, l'enregistrement de son arrivée, la récupération des données logistiques pertinentes qui le concernent et leur ajout dans la base de données.
- Le stockage est l'action de placer le conteneur dans un endroit connu et enregistré de sorte qu'il puisse être récupéré en cas de besoin.
- La préparation concerne surtout les conteneurs qui sont destinés à des navires ou à des trains. Il s'agit d'identifier ces conteneurs et de les organiser de sorte à optimiser les processus de chargement.
- La fonction de chargement consiste à placer les conteneurs appropriés sur les camions, bateaux, ou trains correspondants. Dans cette phase, l'accent est mis sur le contrôle de la chaîne logistique interne du terminal à conteneurs (c'est-à-dire navire->cour->camion(ou train), et respectivement le sens opposé).

2.4.1 Structure de base d'un terminal

D'une manière générale, les terminaux à conteneurs peuvent être décrits comme des systèmes ouverts de flux de matière avec deux interfaces externes. Ces interfaces sont constituées, d'une part des quais avec les chargements (et déchargements) de navires, et d'autre part de la partie terrestre. Une représentation simple d'un terminal à conteneurs est donnée dans la figure 7.

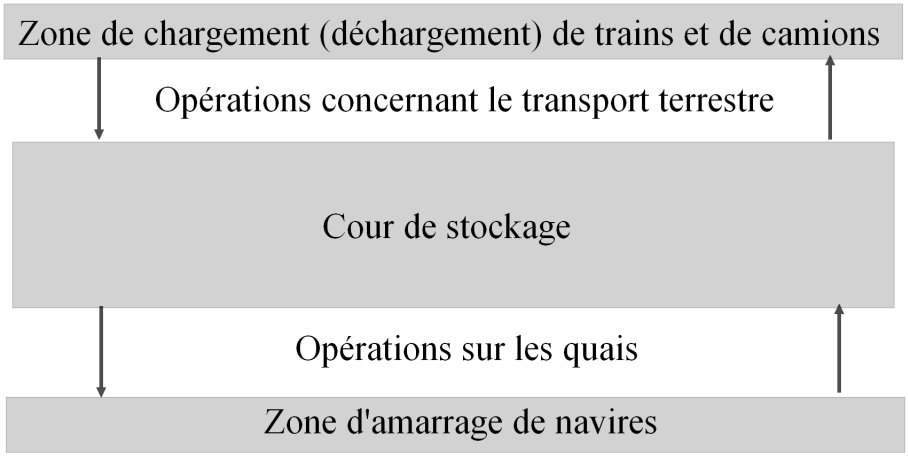


FIGURE 7 – Représentation simple d'un terminal à conteneurs

Pour effectuer des opérations rapides et efficaces, les terminaux à conteneurs disposent d'équipements de manutention et de transfert.

2.4.2 Les équipements de manutention

Généralement, des grues sont utilisées pour effectuer les opérations de manutention. Ces dernières sont principalement réparties entre des grues de quai et des grues de cour.

Les grues de quai servent à charger et à décharger les navires porte-conteneurs. Généralement, le déchargement est réalisé avant le chargement. Plusieurs grues de quai peuvent s'occuper simultanément d'un bateau, mais ils doivent suivre un plan de travail bien déterminé qui assure l'équilibre du navire. Généralement, ces genres de grue circulent sur des rails comme le montre la figure 8.

FIGURE 8 – Grue de quai

Quant aux grues de cour, elles servent à placer (ou à enlever) des conteneurs dans (ou de) la cour de stockage. Ils existent généralement deux types de grue de cour : les "Rail-Mounted Gantry Cranes" (RMGCs) et les "Rubber-Tyred Gantry Cranes" (RTGCs).

- Les RMGCs sont automatisées et ne nécessitent pas de conducteurs. Elles circulent sur des rails, et sont généralement plus rapides que les RTGCs. La figure 9 est un exemple de RMGC.

FIGURE 9 – RMGC

- Les RTGCs (voir Figure 10) ont des pneus en caoutchouc, ce qui leurs donne la flexibilité de circuler librement à l'intérieur de la cour de stockage. Elles sont capables de faire des rotations de 90° pour effectuer des mouvements orthogonaux ; mais ce genre de mouvement nécessite du temps et dure environ 15 minutes [25]. Contrairement à elles, les RMGCs ne peuvent circuler que sur des voies ferrées, mais leur principal avantage est leur rapidité car elles sont capables de parcourir 300m en 1min15sec [102]. En plus de cela, elles peuvent être utilisées à tout moment (sauf

lorsqu'elles sont en panne ou bien en maintenance), contrairement aux RTGCs qui ne sont opérationnelles qu'en présence de main d'œuvre.

FIGURE 10 – RTGC

2.4.3 Les véhicules internes de transfert

Différents types de véhicules sont utilisés pour effectuer les transferts de conteneurs à l'intérieur d'un terminal à conteneurs : des camions, des voitures autoguidés, et des cavaliers gerbeurs.

- Un véhicule autoguidé (AGV) est un robot mobile qui suit des marqueurs ou des fils dans le sol, ou bien qui utilise des aimants ou des lasers pour son orientation. Il est contrôlé par ordinateur, et est doté de pare-chocs automatiques. Chaque AGV est capable de transporter un conteneur, et d'interagir avec des grues. L'utilisation d'AGVs permet essentiellement d'économiser de la main d'œuvre. La figure 11 est un exemple d'AGV.

FIGURE 11 – AGV

- Les cavaliers gerbeurs sont à la fois des véhicules de transport et des matériels de manutention. L'avantage d'un cavalier gerbeur, est le fait qu'il soit capable de soulever un conteneur, de le transporter et de le placer sans aucune intervention. En plus de cela, c'est une machine qui a la possibilité d'empiler jusqu'à quatre conteneurs, et par conséquent, n'a pas besoin de l'intervention de grues. Cependant,

ce genre de véhicule nécessite d'être conduit par un chauffeur (ce dernier s'assoie dans une cabine située tout en haut, de telle sorte qu'il puisse avoir une bonne visibilité du milieu).

FIGURE 12 – Cavalier Gerbeur

Selon Wise et al. [109], une étude menée sur 114 terminaux à conteneurs a révélé que 20.2% d'entre eux utilisent des cavaliers gerbeurs. Cependant, la plupart de ces terminaux à conteneurs (63.2%) utilisent des RTGCs ; contre seulement 6.1% qui utilisent des RMGCs. Les RMGCs sont capables d'interagir directement avec les AGVs, mais pour des raisons de sécurité, leurs échanges avec des camions se font manuellement.

2.4.4 Configurations de la cour de stockage

Dans les terminaux à conteneurs, les espaces de stockage sont constitués de plusieurs blocs. Cependant, les configurations des blocs diffèrent en fonction des équipements de stockage utilisés. On distingue principalement deux types de configuration : le *modèle compact* que l'ont retrouve dans les terminaux à conteneurs qui utilisent des grues de cour, et le *modèle linéaire* que l'ont rencontre dans les terminaux à conteneurs qui se servent de cavaliers gerbeurs.

Dans *le modèle compact*, il n'y a pas d'espaces de séparation prévus entre les piles adjacentes. Un bloc de ce genre est donc constitué de plusieurs rangées qui sont collées les unes aux autres. Chaque rangée est composée de plusieurs travées, qui contiennent à leurs tours des piles dans lesquelles sont superposés des conteneurs.

La disposition de ces blocs par rapport aux quais dépend de la nature des grues de cour utilisées. Dans le cas des terminaux à conteneurs qui ont opté pour des grues non-automatisées (RTGCs), les blocs sont disposés parallèlement aux quais. Dans ces blocs, une ou plusieurs rangées, appelée(s) *voie(s) de camion*, est (sont) réservée(s) à la circulation des véhicules de transfert. De ce fait, ces véhicules circulent dans ces espaces et

s'arrêtent devant les travées souhaitées. Ainsi, les grues se déplacent jusqu'à leurs positions pour effectuer des chargements ou des déchargements. La figure 13 en est une illustration. On retrouve ce genre de disposition dans la plupart des ports asiatiques.

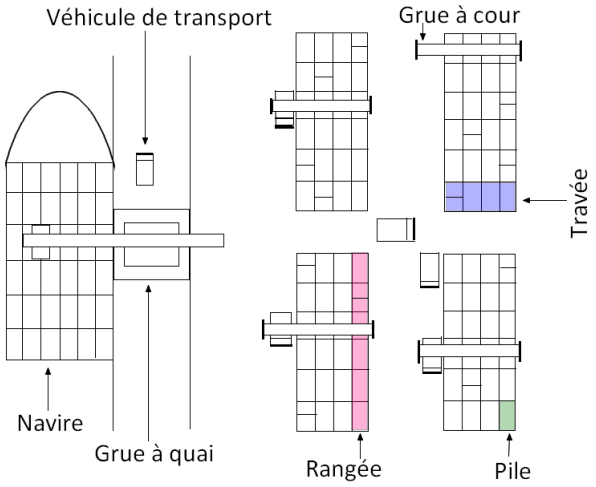


FIGURE 13 – Terminal qui utilise des RTGCs

Dans le cas des terminaux à conteneurs qui utilisent des grues de quais automatisées (RMGCs), les blocs de stockage sont perpendiculaires aux quais. Les échanges entre les véhicules de transfert et les RMGCs se font dans deux zones d'échange spécifiques, situées de part et d'autre de la cour de stockage. Celle qui est en face des quais est le lieu d'inter-action entre les véhicules autoguidés (AGVs) et les RMGCs. Tandis que les échanges entre les camions externes et les RMGCs se font à l'autre côté. Une représentation d'un tel terminal à conteneurs est visible dans la figure 14.

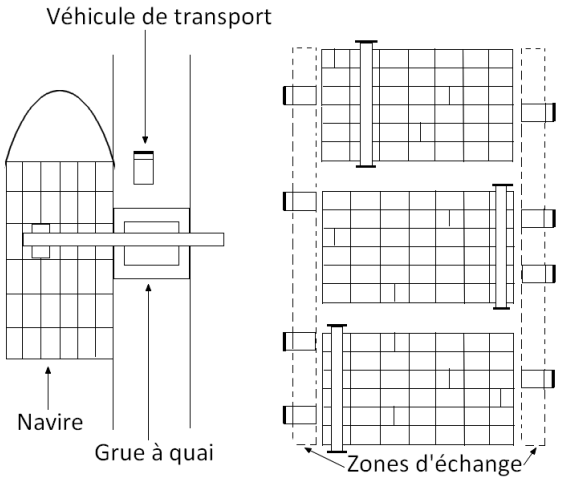


FIGURE 14 – Terminal qui utilise des RMGCs

Dans le cas du *modèle linéaire*, les blocs de stockage sont constitués par des rangées qui ne sont pas collées les unes aux autres. Il y a de petits espaces entre elles, par où

circulent les roues des cavaliers gerbeurs. Chaque rangée est composée de plusieurs piles. La figure 15 en est une illustration, dans laquelle les rangées sont parallèles aux quais ; mais le cas orthogonal est également possible.

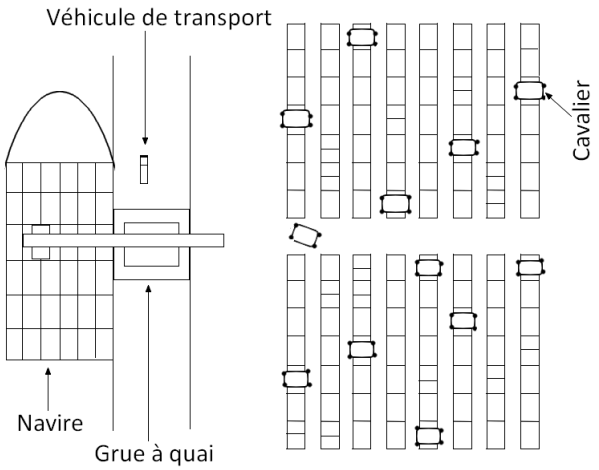


FIGURE 15 – Terminal qui utilise des cavaliers gerbeurs

Pour chacune de ces différentes configurations, les nombres de blocs, de rangées, de travées, et de piles peuvent avoir une influence sur le rendement du terminal. Chu et Huang ont proposé, dans [24], une formule mathématique qui estime le nombre de rangées nécessaire dans des terminaux à conteneurs qui utilisent des cavaliers gerbeurs, ou bien des grues (RTGCs ou RMGCs). Ils ont réalisé une étude sur les terminaux des ports de Taiwan afin de déterminer la meilleure structure pour chaque type d'équipement. Ils ont conclu que : les cavaliers gerbeurs sont plus appropriés aux terminaux qui utilisent un seul quai, alors que les grues de cour sont plus avantageuses pour les grands terminaux.

2.5 Conclusion

De nos jours, les ports occupent une place importante dans la chaîne logistique. La plupart d'entre eux sont devenus des centres de distribution entre plusieurs moyens de transport, c'est leur aspect multi-modal. Cependant, cette place capitale qu'occupent les ports dans le transport et les échanges de biens et de marchandises a été acquise progressivement suite à des améliorations structurelles et fonctionnelles qui se sont imposées. Ces phénomènes sont survenus après la standardisation du conteneur, qui est devenu le principal élément d'échange dans les ports. Ces conteneurs sont gérés par une infrastructure portuaire, appelée terminal à conteneurs, qui s'occupe des processus d'affectation de postes à quai, de chargement, de déchargement, et de stockage.

Bien évidemment, toutes les opérations réalisées dans un terminal à conteneurs doivent être faites de façon efficace, car elles ont des influences sur la productivité générale du port. Cependant, une méthode de gestion optimale de la cour de stockage est difficile à mettre au point à cause de la diversité des données concernant les conteneurs et aussi à des déficits d'informations. Ceci a été à l'origine de nombreux travaux de recherche qui font l'objet du chapitre suivant.

Chapitre 3

Le problème de stockage de conteneurs

3.1 Introduction

La cour de stockage est une ressource capitale dans un terminal à conteneurs. En effet, le degré d'efficacité de sa gestion se répercute dans la productivité globale du port. Suite à la baisse des frais de transport de marchandises et de biens qu'a occasionné la conteneurisation, les flux de conteneurs ont largement augmenté dans les ports. D'ailleurs, selon [105], en 2005 le flux de conteneurs à travers le monde a dépassé 21.6 millions EVPs (équivalent vingt pieds). La conséquence directe de ceci est une croissance vertigineuse du nombre de conteneurs qui séjournent simultanément dans un port, rendant ainsi insuffisants les emplacements de stockage au sol. Une solution de ce problème pourrait être l'augmentation des surfaces des cours de stockage, mais elle s'avère inefficace car les nombres de conteneurs qui entrent simultanément dans un port croissent continuellement en fonction du développement des moyens de transport (exemple : les navires porte-conteneurs). À cela s'ajoute la cherté des parcelles de terrains. Une autre solution à ce problème est d'utiliser l'avantage de la standardisation des tailles des conteneurs, qui rend possible leur superposition. Pour être efficace, la superposition des conteneurs doit aussi tenir compte des dates d'évacuation. En effet, un rangement aléatoire des conteneurs dans une pile peut rendre certains conteneurs inaccessibles lors de leurs extractions. Ainsi, il sera nécessaire d'effectuer des opérations de réarrangement de conteneurs juste avant les heures de départ de certains d'entre eux. Ces opérations sont appelées des *remaniements*, et sont considérées comme des mouvements improductifs qui consomment du temps et qui ralentissent les opérations portuaires. Pour éviter ce genre de désagrément, plusieurs travaux de recherche ont été réalisés ces dernières décennies afin de proposer des méthodes de stockage efficaces et des systèmes d'aide à la décision. Ces derniers font l'objet de ce chapitre, dans lequel sont relatées aussi bien des méthodes analytiques que des méthodes numériques.

Le reste de ce chapitre est organisé comme suit : dans la section 3.2 sont présentées les différentes stratégies de stockage qui sont proposées dans la littérature, la section 3.3 est dédiée aux méthodes de résolution proposées dans la littérature, une conclusion est donnée dans la section 3.4.

3.2 Les différentes stratégies de stockage

Un des avantages de l'utilisation de conteneurs, est le fait qu'il soit possible de les superposer les uns sur les autres. Cependant, cet atout a des limites, dans la mesure où il peut causer des remaniements. En effet, ce genre de manœuvre est surtout effectué lors de l'extraction des conteneurs qui sont aux fonds des piles. Par conséquent, il est capital, pour chaque terminal à conteneurs, d'adopter une stratégie de stockage adéquate. Les différentes méthodes de stockage qui existent dans la littérature peuvent être divisées en quatre catégories selon Saanen et al. [102], qui sont détaillées ci-dessous.

3.2.1 Ségrégation et Non-ségrégation

Le stockage avec *ségrégation* suppose une séparation entre les conteneurs qui sont en exportation (c'est-à-dire les conteneurs sortants) et ceux qui sont en importation (c'est-à-dire les conteneurs entrants). Les conteneurs qui sont destinés à l'exportation sont ceux qui arrivent au port étant chargés sur des camions ou bien sur des wagons, et qui vont quitter le port étant chargés sur des navires. Alors que les conteneurs importés sont amenés au port par des navires, et seront chargés sur des camions ou bien sur des wagons pour être acheminés vers leurs destinations finales. Avec le stockage par ségrégation, les zones de stockage sont préalablement réparties pour déterminer les emplacements qui sont réservés aux conteneurs importés et ceux qui sont destinés aux conteneurs en exportation. Cette répartition peut être faite de trois manières [81]. La première méthode consiste à réserver chaque bloc, soit aux conteneurs importés, soit aux conteneurs en exportation. La deuxième divise en deux parties les rangées de chaque bloc, de ce fait, chaque moitié est réservée à l'une des deux catégories de conteneurs. La troisième méthode est presque semblable à la deuxième. La seule différence est le fait que les divisions se font en considérant les travées. La ségrégation peut même aller jusqu'à subdiviser l'espace réservé à une catégorie de conteneur, par exemple en prédéfinissant la partie réservée à chaque navire. Cette stratégie est surtout utilisée pour le stockage des conteneurs qui vont être chargés sur des navires, d'ailleurs elle est utilisée dans [104] par Ibrahimi et al.

Quant à la méthode de *non-ségrégation*, elle ne fait pas de distinction entre les catégories des conteneurs. Par conséquent, les conteneurs peuvent être superposés indépendamment de leurs destinations.

3.2.2 Groupage et Dispersion

Avec la *méthode de stockage par groupe*, des emplacements (à ne pas confondre avec des piles qui, par définition, en contiennent plusieurs) adjacents sont attribués à chaque ensemble de conteneurs qui ont les mêmes spécificités (exemple : destination, dimension, contenu, frigorifié, vides, etc.). Les conteneurs qui appartiennent à un même groupe sont supposés être interchangeables, et par conséquent, ils peuvent être superposés d'une manière quelconque sans se soucier de leurs dates de départ [40]. La méthode de stockage par groupe est surtout utilisée dans les terminaux à conteneurs qui utilisent des RTGCs ; car, pour économiser de la main d'œuvre, les conteneurs sont regroupés autant que possible afin de minimiser le nombre de grues de cour utilisées [102]. Deux méthodes de

réservation d'emplacements de stockage existent pour cette stratégie [81]. La première méthode appelée "*unité pile*", commence par réserver une pile vide pour chaque groupe de conteneurs, ensuite elle désigne une nouvelle pile vide à chaque fois que celle d'un groupe devient pleine. La deuxième méthode est nommée "*unité travée*" ; elle réserve, dès le début, toute une travée vide à chaque catégorie de conteneur. Et si la travée d'une catégorie devient pleine, elle réserve automatiquement une autre travée vide pour cette catégorie de conteneur.

Contrairement à cette technique, la *méthode du stockage dispersé* n'essaie pas de regrouper les conteneurs. Ces derniers sont stockés indépendamment les uns des autres. Une illustration de cette méthode est le *stockage aléatoire*, qui suppose une équiprobabilité de choix entre les places qui sont compatibles à chaque conteneur [9]. Il peut être résumé comme suit.

1 : Choisir aléatoirement une rangée.
2 : Choisir un emplacement quelconque dans cette rangée.
3 : Tester s'il est possible d'y placer le conteneur.
4 : Si oui, alors effectuer le stockage.
5 : Si non, alors recommencer avec la rangée suivante.

Vue que la méthode aléatoire ne prend pas en considération toutes les informations liées aux conteneurs et aux piles (par exemple : les dates de départ, les distances, etc.), elle ne fournit aucune garantie concernant la qualité des solutions qu'elle procure. Certes, elle fournit des solutions réalisables ; mais il est également intéressant de prendre en considération d'autres paramètres qui peuvent avoir des influences aussi bien sur l'efficacité du stockage que sur celle du déstockage.

3.2.3 Stockage direct et Stockage indirect

Dans la plupart des terminaux portuaires, les conteneurs sont directement placés dans la cour de stockage, où ils vont rester jusqu'à leurs départs. Par contre, dans le cas du *stockage indirect*, les conteneurs sont d'abord placés dans une zone d'agencement avant d'être transférés dans la cour de stockage. L'objectif de ce procédé est de diminuer les temps d'attente des camions externes qui apportent des conteneurs, et aussi d'accélérer les activités des grues de cour en séparant les périodes de stockage et les périodes de retrait. Ainsi, les transferts de conteneurs de la zone d'agencement vers la cour de stockage se font pendant les temps libres des grues de cour. Ce type de procédé est surtout utile pour les ports qui ne disposent pas de toutes les informations concernant les conteneurs à leurs arrivées. De ce fait, les conteneurs sont temporairement mis dans la zone d'agencement, en attendant de récolter toutes les informations qui sont nécessaires pour désigner leurs emplacements dans la zone de stockage finale. Ces informations peuvent être : des dates de départ, des modes de transport (routier ou ferroviaire, dans le cas d'un terminal à conteneurs multi-modal), etc.

L'une des décisions qui peuvent être sujets de réflexion dans les terminaux à conteneurs qui utilisent ce genre de procédé est de trouver la bonne division de l'espace de stockage. Dans [30], Chen a étudié un compromis qui consiste à avoir une zone d'agencement très vaste et une cour de stockage qui a des piles très hautes. Cette solution semble très

convenable aux terminaux à conteneurs qui sont situés dans des pays où les parcelles de terre sont chères.

Certains travaux proposent de ne pas séparer la zone d'agencement et la zone de stockage finale. Par contre, ils considèrent un espace d'agencement à l'intérieur de chaque bloc de stockage. C'est le cas de Castilho et Daganzo, qui, dans [20], se sont intéressés au nombre de mouvements improductifs nécessaires pour retirer des conteneurs qui ne sont pas aux sommets des piles. On retrouve également cette stratégie de stockage dans [104], où les auteurs considèrent que tous les conteneurs qui arrivent dans un bloc sont d'abord placés temporairement dans des piles intermédiaires avant d'être transférés dans les emplacements qui leurs sont dédiés.

En dépit de ses avantages, la méthode de stockage indirect nécessite trop de mouvements qui ne sont pas tous indispensables. Ces déplacements de conteneurs requièrent des moyens techniques (grues ou cavaliers) et aussi de la main d'œuvre dans le cas des terminaux à conteneurs non-automatisés.

3.2.4 Priorité aux déchargements et Priorité aux chargements

Les méthodes de stockage qui priorisent les déchargements de conteneurs cherchent à maximiser les performances de toutes les activités liées aux opérations de stockage. La méthode de stockage par niveau en est une illustration. Elle stocke les conteneurs par couche, de telle sorte que tous les emplacements au sol soient occupés, avant de superposer les conteneurs. Cette stratégie fut proposée par Duinkerken et al. dans [39]. Elle est intuitive mais n'utilise pas la plupart des informations disponibles. Elle contient principalement quatre étapes, qui se succèdent comme suit.

1 : Prendre une rangée quelconque qui a au moins un emplacement libre.
2 : Chercher, dans cette rangée, un emplacement libre et adéquat qui est au contact du sol.
3 : S'il est trouvé : y stocker le conteneur.
4 : S'il n'est pas trouvé : chercher, dans la rangée, un emplacement libre et adéquat qui appartient au niveau le plus bas possible.

Avec le stockage par niveau, le risque de remaniement est moins important qu'avec la méthode aléatoire ; d'ailleurs il peut être inexistant lorsque toute la surface du terminal n'est pas occupée. Cependant, son efficacité est remis en question lorsque le nombre de conteneurs à stocker est largement supérieur à celui des piles. Un autre point faible de cette méthode, est le fait qu'elle puisse nécessiter de parcourir de longues distances dans la cour de stockage, ce qui peut augmenter le temps nécessaire pour effectuer les opérations de stockage ou de dé-stockage.

Contrairement à ces méthodes, celles qui priorisent les chargements de conteneurs ont pour but de maximiser le rendement des opérations de retrait. C'est le cas de la méthode qui stocke les conteneurs suivant l'ordre décroissant de leurs dates de départ [40]. Une version améliorée de cette méthode, appelée stockage par nivellement des dates de départ, est proposée dans [9] par Borgman et al. Leur idée est de stocker les conteneurs suivant l'ordre décroissant de leurs dates de départ, tout en essayant de minimiser la surface

utilisée. Ainsi, si plusieurs piles sont compatibles à un conteneur, alors on le stocke dans celle qui est la plus haute. En plus de cela, le stockage se fait aussi en minimisant les écarts entre les dates de départ de deux conteneurs qui se succèdent dans une même pile. Le nivellement ne se fait donc pas par rapport au sol mais plutôt par rapport aux dates de départ. Avec cette méthode, la recherche d'un emplacement de stockage pour un conteneur se fait en trois étapes qui se succèdent comme suit.

1 : D'abord on cherche parmi les piles qui ne sont ni pleines, ni vides, celles qui ont, à leurs sommets, des conteneurs qui ont des dates de départ supérieures à celle du conteneur que l'on veut stocker. Si on en trouve, on calcule, pour chacune d'elles, la différence entre la date de départ du conteneur qui est à son sommet et celle du conteneur que l'on cherche à stocker. Ensuite on sélectionne la pile qui conduit à la plus petite différence.
2 : Si de telles piles n'existent pas, alors on choisit parmi les piles vides, celle qui est plus proche de la sortie par laquelle le conteneur sera livré.
3 : Si on n'a pas trouvé de pile qui appartient aux deux premiers cas, alors ont stocke le conteneur dans la pile la plus haute parmi celles qui ne sont pas pleines ; afin de minimiser les futurs remaniements.

Selon Borgman et al. [9], cette méthode est plus efficace que le stockage aléatoire et le stockage par niveau, car le risque d'avoir des remaniements est nettement moins élevé. Néanmoins, elle privilégie la minimisation de la surface de stockage utilisée au détriment de la rapidité du stockage ou du dé-stockage qui peuvent avoir des impacts sur la satisfaction des clients.

Des combinaisons entre différentes stratégies de stockage sont proposées dans la littérature, elles seront décrites dans la section suivante.

3.3 Méthodes de résolution

La résolution du problème de stockage de conteneurs se fait généralement en deux étapes. Dans la première phase, des analyses sont effectuées pour rechercher la méthode de stockage adéquate. Ensuite, la partie simulation permet de vérifier et de mesurer l'efficacité des décisions prises dans la partie précédente.

Certains papiers disponibles dans la littérature se sont uniquement focalisés sur l'étude analytique du problème, en proposant des méthodes de stockage pertinentes. Alors que d'autres, en plus de cela, se sont servis de la simulation pour prouver l'efficacité des techniques de stockage qu'ils ont proposées. Dans un premier temps, nous allons parler des publications qui se sont limitées à l'analyse du problème. Après cela, nous décrirons les différentes techniques de simulation ainsi que les algorithmes qui sont proposés dans la littérature pour la résolution du problème de stockage de conteneurs.

3.3.1 Études analytiques du problème de stockage de conteneurs

L'un des premiers articles qui ont proposé des méthodes de stockage basées sur des études analytiques, est celui de Taleb-Ibrahimi [104]. Il s'est intéressé au stockage de

conteneurs qui sont destinés à l'exportation et qui doivent être chargés sur des navires. L'auteur a adopté la stratégie de stockage par ségrégation, en considérant que des emplacements de stockage sont réservés pour chaque navire. Deux variantes de cette méthode de stockage sont relatées dans ce papier : le cas statique et le cas dynamique. Avec la première, il n'est pas permis de changer les emplacements de stockage des conteneurs ; autrement dit, chaque conteneur reste à un endroit fixe durant tout le temps de son séjour. Alors qu'avec la deuxième méthode, des espaces de pré-stockage sont réservés dans chaque bloc. Selon l'auteur, la méthode du stockage dynamique permet une meilleure utilisation de l'espace de stockage, car elle minimise la surface réservée à chaque navire. Il suppose que, lors de l'arrivée du premier conteneur destiné à un navire, les autorités portuaires ne savent pas avec exactitude le nombre de conteneurs qui vont encore arriver et qui sont destinés à ce même navire. De ce fait, le risque encouru avec la méthode statique est d'allouer à un groupe de conteneurs plus d'espace que nécessaire. Cette situation, ne risque pas de se produire avec la méthode dynamique, car les conteneurs ne sont transférés à leurs emplacements finaux qu'après un temps limite de réception de conteneurs ; ainsi la surface de stockage nécessaire est connue avec précision. Taleb-Ibrahimi a considéré un terminal à conteneurs qui utilise des grues de cour, et a proposé des formules mathématiques pour calculer le nombre d'emplacements nécessaire pour stocker les conteneurs, aussi bien pour le cas statique que pour le cas dynamique. Il a également proposé des procédures pour déterminer les moments idéaux pour réserver, à chaque navire, des espaces dans les zones de stockage permanent. Cependant, il a restreint son étude, en considérant uniquement le cas du stockage où tous les conteneurs ont les mêmes dimensions ; ce qui n'est pas toujours le cas dans la vie réelle, car des conteneurs de tailles variées peuvent être chargés simultanément sur un navire.

Le cas des conteneurs importés (c'est-à-dire qui sont déchargés des navires) est traité dans [20] par Castilho et Daganzo. Ils ont considéré un terminal à conteneurs qui utilise des grues de cour, et ont proposé des formules mathématiques pour calculer le nombre de remaniements nécessaire pour extraire un conteneur de la cour de stockage. Dans cet article, le nombre de remaniements est considéré comme étant égal au nombre de conteneurs qui sont au-dessus du conteneur ciblé. Les auteurs ont analysé deux méthodes de stockage différentes. La première vise à équilibrer les hauteurs des piles (c'est une stratégie de nivellement) ; de ce fait, les conteneurs déplacés sont transférés dans les piles les moins remplies et qui ne sont pas très loin de leurs emplacements initiaux. Castilho et Daganzo ont remarqué que cette méthode conduit à plus de remaniements que la stratégie aléatoire qui ne tient pas compte de la hauteur des piles. En plus de cela, ils ont aussi constaté que l'inconvénient de cette méthode est le risque de placer des conteneurs fraîchement arrivés au terminal sur d'autres qui ne vont pas tarder à partir. Ainsi, pour palier à cela, ils ont proposé une stratégie de ségrégation qui alloue à chaque navire une zone de stockage. La partie réservée à chaque navire est divisée en plusieurs groupes de piles dans lesquelles les conteneurs peuvent être remaniés (c'est-à-dire que certains conteneurs peuvent être enlevés de leurs emplacements initiaux pour être placés dans d'autres groupes de piles qui sont dans la même zone). Avec cette méthode, il peut arriver une situation dans laquelle l'annonce de l'arrivée d'un navire coïncide avec un moment où toute la surface du terminal est occupée. Dans ce genre de situation, les auteurs proposent de vider les groupes les moins remplis en transférant leurs conteneurs dans d'autres groupes, tout en s'assurant que chaque conteneur déplacé est mis dans un autre emplacement qui est

également réservé à son navire d'origine. Castilho et Daganzo ont remarqué que cette dernière stratégie conduit à moins de remaniements ; car les conteneurs se trouvent souvent dans des piles qui sont presque vides au moment de leur réclamation. Mais, vu que le fait de déplacer des conteneurs de certains endroits à d'autres peut nécessiter de parcourir de longues distances, cette stratégie semble plus performante dans le cas d'un terminal à conteneurs qui n'a pas une surface de stockage assez grande, et qui autorise des piles très hautes. Les auteurs ont considéré, dans leurs calculs, que les navires arrivent au port dans des intervalles de temps réguliers, et que les durées de séjour des conteneurs suivent une loi de distribution uniforme. Par conséquent, même si la stratégie qu'ils ont proposée semble très attirante, on a aucune garantie sur son efficacité dans d'autres situations.

Dans la majorité des terminaux à conteneurs, les conteneurs ont des durées de séjour variables. Cependant, avec l'agrandissement continuel des navires porte-conteneurs, les terminaux sont de plus en plus confrontés à la nécessité d'une bonne gestion de la cour de stockage et des équipements de manutention. Lorsque les durées de séjour des conteneurs au port sont trop longues, le risque encouru est un encombrement de la cour de stockage, ce qui est néfaste car pouvant provoquer des pénuries d'espace et des difficultés à accéder à certains conteneurs. Le risque d'avoir ce genre de situation est plus élevé avec le cas des conteneurs importés qui sont généralement réclamés individuellement par des camions. Ainsi, pour encourager les propriétaires à ne pas laisser leurs conteneurs au port pendant des durées indéterminées, certains auteurs ont proposé de fixer une durée de stockage gratuit au delà de laquelle il y a des frais de stockage. C'est le cas de Kim et Kim, qui ont proposé dans [63] trois modèles mathématiques pour déterminer la durée de stockage gratuit. Dans le premier, ils ont considéré un terminal à conteneurs qui appartient à une entreprise privée dont le seul but est de maximiser ses bénéfices sans se soucier des conséquences. Selon les auteurs, ce genre de port a plutôt tendance à proposer une longue durée de stockage gratuit et de bas prix pour les durées de stockage additionnelles, afin d'empocher les frais de stockage que les clients pourraient payer à d'autres entrepôts externes. Mais, ils ont souligné que les inconvénients d'un tel procédé sont : l'encombrement de la cour de stockage, la difficulté d'accéder à certains conteneurs, et par conséquent l'allongement des durées d'attente des camions externes qui récupèrent des conteneurs. Dans le deuxième modèle, les auteurs considèrent un terminal à conteneurs qui cherche à maximiser ses profits tout en essayant de limiter les durées d'attente des camions lorsqu'ils viennent récupérer des conteneurs. Selon eux, l'avantage de cette méthode est qu'elle permet au terminal à conteneurs d'être compétitif par rapport aux autres, tout en faisant des bénéfices. Dans le troisième modèle, les auteurs ont considéré un terminal à conteneurs publique (c'est-à-dire qui est construit grâce à des fonds publiques et qui est géré par un État ou bien un gouvernement) qui ne cherche qu'à maximiser les profits de ses clients. Après avoir étudié ce dernier modèle, les auteurs sont arrivés à la conclusion qu'il vaut mieux pour ce genre de port d'éliminer la période de stockage gratuit, afin d'inciter les propriétaires des conteneurs à récupérer leurs biens très vite, comme ça la cour de stockage sera moins encombrée ; et, par conséquent, les durées d'attente des camions seront minimales. Dans tous ces trois modèles, les auteurs ont considéré que les manutentions sont effectuées par des grues, et que des conteneurs qui viennent de différents navires ne peuvent pas être stockés dans une même travée (c'est une stratégie classique de ségrégation).

3.3.2 Études s'appuyant sur la simulation pour résoudre le problème de stockage de conteneurs

Grâce au développement croissant et continuel de la technologie, l'ordinateur devient de plus en plus sophistiqué. L'une des principales utilités de cette machine est le fait qu'elle soit capable d'effectuer rapidement des tâches qui pourraient nécessiter beaucoup de temps à l'homme. En plus de cela, la diversité des langages de programmation fait de lui un outil très convoité pour effectuer des calculs ou bien même des simulations numériques. Ces atouts sont de plus en plus exploités pour résoudre le problème de stockage de conteneurs dans un terminal portuaire.

Kap Hwan Kim [59] est l'un des premiers à coder un programme informatique en langage C pour calculer le nombre total de remaniements prévu lors de l'extraction des conteneurs d'un bloc. Dans cet article, l'auteur s'est uniquement intéressé au stockage des conteneurs qui sont apportés par des navires (c'est-à-dire les conteneurs qui sont importés), et a considéré un terminal à conteneurs qui utilise des grues de cour. Cependant, il s'est appuyé sur une méthode de stockage par ségrégation qui ne permet pas de mélanger des conteneurs venant de navires différents dans une même travée. En plus de cela, il a supposé que les conteneurs déplacés lors d'un remaniement seront stockés dans la même travée où se situent leurs emplacements initiaux ; de ce fait, il ne sera pas nécessaire de déplacer la grue de cour. Les principales contributions de ce papier sont des tableaux et des équations qui servent à calculer le nombre de remaniements encourus lors de l'extraction des conteneurs par les grues de cour suivant un ordre aléatoire. Mais le degré de précision de ces méthodes de calcul reste à déterminer car l'auteur a considéré qu'à chaque instant tous les conteneurs ont les mêmes probabilités d'être réclamés, ce qui ne correspond pas tout à fait à la réalité.

Un autre avantage de la simulation est le fait qu'elle permette de réaliser plus facilement des études quantitatives. Sculli et Hui [91], s'en sont servi pour étudier un cas particulier du problème de stockage de conteneurs, où tous les conteneurs ont les mêmes dimensions. Ils ont basé leur étude sur l'observation des variations de quatre critères de performance : le taux de remplissage de la cour de stockage, le nombre de mouvements improductifs, le nombre de conteneurs qui n'ont pas pu être stockés, et aussi le nombre de places non occupées. Les auteurs ont considéré que chaque pile peut contenir au maximum trois conteneurs ; par conséquent, le volume d'un bloc est égal au nombre d'emplacements au sol multiplié par 3. Le nombre de mouvements improductifs est égal au nombre de déplacements de conteneurs non réclamés ; ce genre de mouvements survient lors de l'extraction d'un conteneur qui est en-dessous de d'autres conteneurs. Les différences entre les instances qu'ils ont utilisées dans leurs tests se trouvent aux niveaux de : la taille maximale de l'espace de stockage considéré, la règle de stockage appliquée, et le nombre de types de conteneur. Deux stratégies de stockage sont évaluées dans ce papier : une méthode aléatoire et une stratégie de stockage par catégorie. Avec la première méthode, chaque conteneur est affecté au premier emplacement trouvé, en commençant d'abord par ceux qui sont au niveau 1 (au sol), puis ceux du niveau 2, et enfin ceux du troisième niveau s'il n'y a plus de places dans les deux précédents. La deuxième méthode prend en considération les types des conteneurs en favorisant le regroupement de conteneurs de même catégorie dans chaque pile. L'inconvénient de cette stratégie est le fait qu'il peut arriver une situation où il est impossible de trouver une place qui est appropriée à un

conteneur donné, même si la cour de stockage n'est pas pleine ; dans ce cas, les auteurs proposent de stocker le conteneur suivant la méthode aléatoire. Après avoir effectué des simulations sur trente deux différentes instances, Sculli et Hui ont remarqué que ces deux stratégies n'affectent pas le pourcentage de remplissage de la cour de stockage, par contre elles influent différemment sur le nombre de mouvements improductifs. Selon eux, la méthode aléatoire conduit à des piles très hautes, en plus de cela, contrairement à l'autre méthode, elle engendre aussi beaucoup de mouvements improductifs. L'absence de remaniements dans la deuxième stratégie de stockage sous entend que les conteneurs d'une même catégorie seront réclamés simultanément. Le nombre de types de conteneur ainsi que la capacité maximale de la cour de stockage ont eux aussi des impacts sur le nombre de remaniements. Une autre révélation de leurs expériences est le fait que le nombre de conteneurs non stockés et le nombre de places restantes dépendent uniquement de la capacité maximale de l'espace total de stockage, d'ailleurs ils décroissent lorsque la taille du terminal augmente. Bien vrai que Sculli et Hui ont pris en considération, dans ce papier, des paramètres pertinents pour la résolution du problème de stockage de conteneurs, cependant ils ont trop simplifié le problème en considérant que tous les conteneurs ont les mêmes dimensions et que les conteneurs de même type partiront simultanément, ce qui n'est pas toujours le cas dans les terminaux à conteneurs.

Dans [81], Ma et Kim ont utilisé le eM-plant, qui est un logiciel d'optimisation, pour comparer les performances de différentes stratégies de stockage. Ils ont considéré un terminal à conteneurs qui utilise des grues automatisées (RMGCs) et des camions, et dont les blocs sont disposés parallèlement aux quais. Les performances étudiées dans cet article sont : les temps que passent les camions dans les blocs, la durée des trajets parcourus par les camions entre les quais et la cour de stockage, et le nombre de conteneurs qui n'ont pas obtenu de places de stockage. Les simulations relatées dans ce papier, concernent aussi bien les conteneurs qui sont importés que ceux qui sont en exportation. Cependant, ces conteneurs ne sont pas mélangés entre eux dans la cour de stockage. Trois méthodes de ségrégations ont été testées dans cette étude. La première, nommée "*REBLOCK*", associe chaque bloc à un type de conteneur. La deuxième, appelée "*RSROW*", divise les rangées de chaque bloc en deux parties, dont l'une est réservée aux conteneurs importés et l'autre aux conteneurs qui sont en exportation. La troisième méthode, désignée par "*RSBAY*" sépare les travées de chaque bloc en deux groupes égaux qui sont chacun destinés à un type de conteneur. En plus de cette séparation, les auteurs ont choisi de stocker les conteneurs qui sont en exportation par groupes selon leurs dimensions et leurs navires de destination. Cependant, pour allouer des espaces de stockage à ces groupes, deux règles sont testées : celle qui alloue une pile par demande ("*unité pile*"), et celle qui réserve une travée par demande ("*unité travée*"). Après avoir effectué plusieurs simulations, Ma et Kim ont remarqué qu'avec la méthode de stockage du "*REBLOCK*", les camions passent plus de temps dans les blocs. Néanmoins, cette méthode ainsi que le "*RSBAY*" et le "*RSROW*", conduisent à plus de pénuries de place qu'avec la méthode de stockage aléatoire qui ne fait pas de restriction sur les natures des conteneurs. De même, la méthode du "*unité travée*" entraîne plus de manques de place que la méthode du "*unité pile*", mais son avantage est que le temps moyen passé par les camions dans les blocs est moins important. Les auteurs ont aussi constaté que le fait de privilégier les blocs qui sont proches des quais est moins avantageux que la méthode du choix aléatoire de bloc, en terme de temps moyen passé par les camions à l'intérieur des blocs.

Jiang et al. ont proposé dans [56] un système, codé en C++, qui contient à la fois une méthode de répartition de la cour de stockage et une technique d'attribution des charges de travail aux grues de cour. Ce système, qui est spécialement conçu pour la résolution du problème de stockage de conteneurs qui sont en transition entre des navires différents, progresse en deux phases successives. Dans la première phase, il alloue des zones de stockage à chaque navire (cette étape est appelée "*template generation*"). Après cela, il précise, dans la deuxième phase appelée "*space allocation and workload assignment*", les espaces de stockage qui sont communs à deux navires différents ainsi que le nombre de conteneurs affectés à chaque grue de cour. Trois modèles mathématiques ont été proposés pour l'implémentation de ces deux phases, dont l'un pour le "*template generation*" et les deux autres pour le "*space allocation and workload assignment*". L'objectif visé par les auteurs, dans le premier modèle mathématique, est d'allouer aux navires des espaces de stockage qui ont des tailles minimales mais suffisantes. En ce qui concerne les deux modèles proposés pour la résolution du "*space allocation and workload assignment*", ils illustrent chacun une variante de la stratégie de ségrégation. Ces deux variantes sont fondées sur la même idée de base qui consiste à séparer, par un espace de stockage commun, deux espaces consécutifs qui sont réservés à deux navires différents. Cependant, la différence entre ces deux nouvelles stratégies est que dans la première (nommée "*fixed shared space*") les espaces communs ont des mesures égales ; alors que dans la deuxième (appelée "*variable sharing space*") les espaces communs ont différentes mesures qui sont déterminées en fonction des charges de travail des grues à cour (notons que dans cet article, une grue est affectée à chaque navire). L'objectif recherché dans le modèle qui est proposé pour la stratégie de "*fixed shared space*" est de maximiser la taille totale des espaces communs, alors que celui convoité dans le modèle qui est spécifique à la stratégie de "*variable sharing space*" est de minimiser la taille totale des espaces de stockage réservés dans chaque bloc. Des simulations numériques ont été réalisées par Jiang et al. pour la validation du système qu'ils ont proposé. Ces dernières ont révélé que les deux nouvelles stratégies sont meilleures que la stratégie classique de ségrégation (qui réserve un espace fixe de dimension invariable à chaque navire), mais elles n'ont pas les mêmes performances car la stratégie du "*variable sharing space*" donne les meilleurs résultats. Cependant, Jiang et al. ont souligné qu'ils n'ont pas pris en compte, dans les modèles mathématiques qu'ils ont proposés, des incertitudes qui peuvent concerner les données qu'ils ont utilisées, et ont suggéré d'étendre leur travail à ce niveau.

Jusqu'ici, les travaux relatés n'incluaient pas de techniques d'optimisation précises. Cependant, des publications apparues au cours de ces dernières décennies ont prouvé l'utilité de la recherche opérationnelle dans la résolution du problème de stockage de conteneurs. Certaines des méthodes d'optimisation rencontrées dans la littérature, fournissent des résultats optimaux. Mais le principal point faible de ces méthodes exactes est le fait qu'elles nécessitent beaucoup de temps de calcul, dans la plupart des cas. Ainsi, pour remédier à cela, des algorithmes heuristiques et méta-heuristiques sont créés. Ces derniers nécessitent généralement des temps de calcul raisonnables, mais l'optimalité des solutions qu'ils procurent n'est pas garantie. Les algorithmes heuristiques, proposés dans la littérature, sont généralement spécifiques à des cas particuliers du problème. Alors que les algorithmes méta-heuristiques sont plus flexibles, les plus connus d'entre eux sont : la méthode "tabou", le recuit simulé, les algorithmes génétiques, et l'algorithme de colonie

de fourmis. Dans les paragraphes qui suivent, nous allons parler de ces techniques d'optimisation, tout en citant des références dans lesquelles elles sont utilisées pour résoudre le problème de stockage de conteneurs.

3.3.3 Programmation dynamique

La programmation dynamique est créée en 1957 par Richard Bellman [17]. Selon Culioli [26] (page 295) : "*c'est une méthode d'optimisation dynamique particulièrement bien adaptée aux problèmes d'optimisation séquentiels, c'est à dire pour lesquels on désire minimiser un coût séparable en temps, le long d'une trajectoire*".

Cette méthode fait intervenir naturellement une variable de temps qui est aussi appelée variable d'étape, notée t (qui peut être un nombre ou bien un intervalle compris dans [0,T]), une variable d'état appelée x (avec $x(t) \in \mathbb{R}^n$), et une variable de décision nommée u (avec $u \in \mathbb{R}^m$). L'évolution d'un système dynamique est décrite par une *équation d'état*, dont la formule est : $x(t+1) = f(x(t), u(t), t)$.

Ainsi la connaissance de l'état du système $x(t)$ en un instant donné et d'une suite de décision $u(t+k)$, $k \geq 0$ et $t+k \leq T-1$, conduit à la connaissance de tous les états successifs. Cela est en parfait accord avec le principe d'optimalité de Bellman selon lequel : "*dans un processus d'optimisation dynamique, une suite de décisions est optimale si, quels que soient l'état et l'instant considérés sur la trajectoire qui lui est associée, les décisions ultérieures constituent une suite optimale de décisions pour le sous-problème dynamique ayant cet état et cet instant comme conditions initiales*" [26] (page 297).

Dans le cas continu, l'*équation d'état* devient : $\dot{x} = f(x(t), u(t), t)$.

Différents papiers concernant la résolution du problème de stockage de conteneurs ont utilisé la programmation dynamique. Dans [60], Kim et Bae ont utilisé cette méthode pour résoudre le problème de relocalisation de conteneurs dans un terminal portuaire. Ils se sont particulièrement intéressés aux conteneurs qui vont être chargés sur les mêmes navires. Selon eux, le fait de changer les emplacements des conteneurs permet d'accélérer les chargements des navires. Ils supposent que le plan de chargement d'un navire n'est généralement connu qu'après les arrivées et les stockages de tous les conteneurs qui lui sont destinés. Par conséquent, il peut arriver que l'ordre de chargement de ces conteneurs sur les navires ne correspond pas à l'ordre dans lequel ils sont disposés dans la cour de stockage. Ainsi, pour convertir une disposition donnée de conteneurs en une disposition souhaitée, Kim et Bae ont proposé une méthodologie qui a pour objectif de déplacer le moins de conteneurs possible et de minimiser les distances parcourues. Pour ce faire, ils ont décomposé le problème en trois sous-problèmes qui sont : le problème de correspondance entre travées, la planification des déplacements, et l'ordonnancement des tâches. Le problème de correspondance entre travées consiste à associer chaque travée de la disposition initiale à une autre travée dans la configuration finale souhaitée. Dans l'étape de la planification des mouvements, les nombres de conteneurs à transférer entre deux travées quelconques sont déterminés. Le troisième sous-problème consiste à minimiser le temps total des opérations de relocalisation. Les auteurs ont commencé par résoudre simultanément le problème de correspondance entre travées et le problème de planification des déplacements. A chaque itération, ils utilisent la programmation dynamique pour résoudre le premier sous-problème, ensuite ils se servent des résultats obtenus pour ré-

soudre le deuxième sous-problème en utilisant une méthode de résolution de problème de transport. Mais, vu qu'ils ont considéré que deux grues de cour ne peuvent pas travailler simultanément à moins qu'une distance de sécurité qui est égale à la largeur de deux travées les sépare, certaines correspondances entre travées peuvent poser des problèmes, et par conséquent elles seront ajoutées à une liste d'interdictions. Ensuite le problème de correspondance est résolu une autre fois en tenant compte des correspondances interdites. Ces deux procédures sont répétées jusqu'à l'obtention d'une planification correcte de mouvements. Après cela, le troisième sous-problème (c'est-à-dire le problème d'ordonnancement des tâches) est résolu, sous forme d'un problème de voyageur de commerce contenant des relations de précédence exprimées sous forme de conditions de satisfaction, grâce à la programmation dynamique. Les auteurs ont remarqué que les techniques de programmation mathématique qu'ils ont proposées nécessitent beaucoup de temps de calcul, surtout pour l'ordonnancement des tâches, et ont suggéré l'utilisation d'algorithmes heuristiques.

Dans [62], Kim et al. ont proposé un algorithme de programmation dynamique pour allouer des emplacements de stockage aux conteneurs qui sont en exportation. Ils ont considéré un terminal à conteneurs qui utilise des camions et des grues de cour comme équipement de manutention. Cependant, les auteurs ont pris en compte les différences de poids entre les conteneurs, dans l'approche qu'ils ont proposée pour la résolution du problème. Selon eux, pour assurer l'équilibre d'un navire, les conteneurs les plus lourds devront être placés au fond. Par conséquent, pour accélérer le chargement des navires, ils ont proposé de disposer les conteneurs dans la cour de stockage d'une manière telle que les conteneurs les plus lourds soient aux sommets des piles. Leur objectif principal est de minimiser le nombre de remaniements prévu lors des chargements des navires, même s'ils ont préalablement supposé que les conteneurs qui sont affectés à une travée appartiennent à un même groupe (même destination et même type). Cependant ils ont considéré que les camions externes qui amènent ces conteneurs sont déchargés suivant la règle du premier-arrivé/premier-servi. En plus de cela, chaque conteneur ne peut être déplacé vers un autre emplacement de stockage qu'une seule fois. Après avoir testé le modèle de programmation dynamique qu'ils ont proposé pour résoudre ce problème de stockage de conteneurs, Kim et al. ont constaté que ce dernier nécessite des temps de calcul très longs et qu'il n'est pas approprié pour aider à la prise de décision en temps réel. Ainsi, pour diminuer les temps de résolution, les auteurs ont construit un arbre de décision dont les nœuds sont des solutions qui sont obtenues à partir de leur modèle de programmation dynamique. Par contre, l'optimalité des solutions que procure l'arbre de recherche n'est pas garantie, puisque les auteurs ont utilisé dans le parcours de ce dernier des procédures de classification qui contiennent : des critères de sélection d'attributs clés pour faire des branchements, une règle d'élagage, et une méthode de simplification. D'ailleurs, les comparaisons effectuées par les auteurs, entre les résultats obtenus par leur arbre de recherche et leur programme dynamique, ont révélé que le pourcentage de solutions non-optimales fournies par l'arbre de recherche varie entre 1.0% et 5.5% en fonction de la grandeur de l'erreur tolérée lors des processus d'élagage.

3.3.4 Méthode du sous-gradient

La méthode du sous-gradient est un algorithme simple pour minimiser (ou maximiser) une fonction non différentiable qui est soit convexe (dans le cas d'une minimisation) ou bien concave (dans le cas d'une maximisation). Elle a été créée vers les années 1970 en Union Soviétique par Shor [95]. Cependant, elle ressemble beaucoup à la méthode du gradient qui est spécifique aux fonctions différentiables, mais il y a plusieurs différences entre ces deux méthodes. Par exemple, la méthode du sous-gradient utilise des "*pas de déplacement*" qui ont des longueurs fixes au cours du temps, au lieu d'une ligne de recherche exacte ou approximative comme c'est le cas dans une méthode de gradient. Pour bien comprendre cette méthode, il est important de connaître la définition d'un sous-gradient, qui est la suivante.

Soit une fonction convexe $f : \mathbb{R}^n \to \mathbb{R}$, et $x \in \mathbb{R}^n$.
Un sous-gradient de f en x est un vecteur quelconque g qui satisfait l'inégalité suivante.

$$f(y) \geq f(x) + g^T(y - x), \quad \forall y \in \mathbb{R}^n$$

Si f est différentiable alors l'ensemble des sous-gradients de f est réduit à son gradient.
Pour minimiser la fonction f avec la méthode du sous-gradient, on utilise la formule suivante

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)},$$

où x^k est calculé à la $k^{\text{ème}}$ itération, $g^{(k)}$ est un sous-gradient quelconque de f en $x^{(k)}$, et $\alpha_k > 0$ est le *"pas de déplacement"* à l'étape k. Ainsi, à chaque itération de l'algorithme de la méthode du sous-gradient, on avance d'un pas dans le sens contraire du sous-gradient.
Puisque cette méthode n'est pas une méthode de descente, il est utile de mettre à jour la meilleure solution trouvée au fur et à mesure que l'algorithme progresse. À chaque itération k, on pose

$$f^{(k)}_{meilleur} = \min\{f^{(k-1)}_{meilleur}, f(x^{(k)})\},$$

et on pose $i^{(k)}_{meilleur} = k$ si $f(x^{(k)}) = f^{(k)}_{meilleur}$, autrement dit si $x^{(k)}$ est le meilleur point trouvé depuis le début jusqu'à l'itération actuelle. (Dans une méthode de descente, il n'est pas utile de faire cela, car à chaque itération la nouvelle solution calculée est meilleure que celles des itérations précédentes). On a donc

$$f^{(k)}_{meilleur} = \min\{f(x^{(1)}), ..., f(x^{(k)})\},$$

ce qui signifie que $f^{(k)}_{meilleur}$ est la valeur de la meilleure solution trouvée en k itérations.
Ils existent différentes règles pour fixer le "*pas de déplacement*" (α_k) dans la méthode du sous-gradient, les plus connues sont détaillées ci-dessous.

- *Taille de pas constante* : $\alpha_k = h$, où h est une constante indépendante de k.
- *Longueur de pas constante* : $\alpha_k = \frac{h}{||g^{(k)}||_2}$.
- *Carré sommable* : α_k doit satisfaire la condition

$$\sum_{k=1}^{+\infty} \alpha_k^2 < +\infty, \quad \sum_{k=1}^{+\infty} \alpha_k = \infty.$$

Un exemple typique qui utilise cette règle est $\alpha_k = a/(b+k)$, où $a > 0$ et $b \geq 0$.

- *Diminution non sommable* : α_k vérifie la condition

$$\lim_{k \to +\infty} \alpha_k = 0, \quad \sum_{k=1}^{+\infty} \alpha_k = +\infty.$$

Un exemple de pas appartenant à cette catégorie est $\alpha_k = a/\sqrt{k}$, avec a>0.

Ils existent plusieurs résultats sur la convergence de la méthode du sous-gradient. Pour les cas de "*pas de déplacement*" qui concordent avec les deux premières règles (*Taille de pas constante* et *Longueur de pas constante*), il est démontré dans [19] que la méthode du sous-gradient converge vers une solution qui est très proche de l'optimum global, ce qui signifie qu'on a

$$\lim_{k \to +\infty} f^k_{meilleur} - f^* < \epsilon,$$

où f^* est la valeur de la solution optimale du problème, autrement dit $f = \inf_x f(x)$. Ce qui signifie que la méthode du sous-gradient trouve une solution ϵ-sous-optimale après un nombre fini d'itérations. La valeur de ϵ dépend de celle du paramètre h, d'ailleurs elles décroissent simultanément dans le cas de la règle du *longueur de pas constante*.

En ce qui concerne les deux dernières règles (*carré sommable* et *diminution non sommable*), il a également été démontré dans [19] que l'algorithme de la méthode du sous-gradient converge vers la solution optimale, ce qui signifie qu'on a

$$\lim_{k \to +\infty} f(x^{(k)}) = f^*.$$

Si la fonction f est différentiable, alors la première règle (*Grandeur de pas constant*) conduit à une convergence vers la solution optimale si la valeur de h est assez petite.

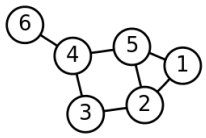
Beaucoup de recherches se sont appuyées sur la méthode du sous-gradient pour résoudre des problèmes concrets de la vie réelle. Dans [61], Kim et Park ont proposé deux algorithmes heuristiques pour résoudre le problème de stockage de conteneurs. L'un est basé sur la méthode du sous-gradient, tandis que l'autre algorithme qu'ils ont nommé *Least duration-of-stay rule* se fonde sur les politiques de stockage partagés. Les auteurs se sont intéressés à la résolution du cas dynamique du stockage des conteneurs qui sont en exportation (c'est-à-dire qui vont être chargés sur des navires). Pour ce faire, ils ont considéré un horizon de planification qu'ils ont divisé en plusieurs étapes. Ainsi, à chaque période du planning, ils connaissent le nombre de conteneurs en exportation qui viennent d'arriver au port et le nombre de navires qui quittent le port. De ce fait, en résolvant le problème, ils tiennent compte des variations de l'état de la cour de stockage (autrement dit des nombres d'emplacements occupés, réservés, et libres). Cette méthode est appelée le "*rolling horizon approach*". Dans ce papier, Kim et Park ne déterminent pas l'emplacement exact alloué à chaque conteneur, mais ils calculent, pour chaque période et pour chaque bloc, le nombre de places allouées aux conteneurs destinés à chaque navire. Par conséquent, ils n'ont pas fait allusion à une stratégie de stockage précise, mais leur principal objectif est de minimiser le coût total des transports de conteneurs entre la cour de stockage et les quais lors des chargements des navires. Cependant, ils ont considéré deux types de transferts : l'un est direct, et l'autre est indirect. Dans le transfert direct, les

transports ainsi que les stockages et les extractions des conteneurs sont assurés par des cavaliers gerbeurs (ce sont des engins qui sont capables de soulever, de transporter, et de déposer des conteneurs). Alors que dans le transfert indirect, les conteneurs sont placés (ou retirés) dans la cour de stockage par des grues. Kim et Park ont proposé un modèle mathématique général de programmation mixte en nombres entiers pour ces deux types de transfert. Ce modèle a ensuite été allégé suivant la méthode de relaxation lagrangienne, après cela, il a été décomposé, avant d'être résolu avec une méthode de sous-gradient. En ce qui concerne l'algorithme du *Least duration-of-stay rule*, il est myope, car il alloue séquentiellement des emplacements aux navires sans faire attention aux conséquences que peut avoir l'allocation faite pour un navire sur les futures allocations concernant d'autres navires. Il commence par allouer l'espace le moins coûteux au navire qui a la date de départ la plus proche. L'algorithme progresse étape par étape durant tout l'horizon de planification. À chaque étape, les demandes d'emplacements seront séquencées suivant l'ordre croissant des dates de départ de leurs correspondants navires. L'espace le moins coûteux est ensuite alloué à la première demande non encore satisfaite de la séquence. La procédure d'affectation est répétée jusqu'à la satisfaction de toutes les demandes. Après avoir effectué des simulations numériques, Kim et Park ont constaté que l'algorithme heuristique du sous-gradient donne des résultats qui sont meilleurs que ceux de l'algorithme du *Least duration-of-stay rule*, mais ce dernier est plus rapide.

3.3.5 Utilisation de la théorie des graphes

La théorie des graphes est utilisée sous différents aspects pour résoudre divers problèmes concrets, notamment celui du stockage de conteneurs dans un terminal portuaire. Les définitions qui suivent sont utiles pour faciliter la compréhension des différentes méthodes de résolution de ce problème, qui sont proposées dans la littérature et qui sont basées sur des notions de graphe.

Graphe : on appelle graphe un couple ordonné $G = (V, E)$, comprenant un ensemble de sommets noté V et un ensemble d'arêtes (dans le cas d'un graphe non-orienté, ou d'arcs dans le cas d'un graphe orienté) dénommé E. Chaque élément de E est un couple d'éléments de V, comme on peut le voir dans la figure 16.



$V = \{1; 2; 3; 4; 5; 6\}$, et $E = \{(6, 4); (4, 5); (4, 3); (3, 2); (5, 2); (5, 1); (2, 1)\}$

FIGURE 16 – Exemple de graphe

Des informations supplémentaires peuvent être contenues dans le graphe. Dans le cas d'un graphe orienté, la capacité de chaque arc peut être renseignée ; celle-ci est une fonction définie comme suit

$$c : E \to \mathbb{R}^+.$$

Le graphe orienté devient alors un réseau qui est noté $G = (V, E, c)$. Considérant un tel graphe, on peut définir la notion de *flot*.

Flot dans un graphe : soit $G = (V, E, c)$ un graphe ayant un seul sommet source s, et un seul sommet puits p. Un flot défini de s à p est une fonction $f : E \to \mathbb{R}$, qui vérifie les deux conditions suivantes

$$\sum_{(i,j)\in E} f(i,j) = \sum_{(j,k)\in E} f(j,k), \quad \forall j \in V \setminus \{s,p\} \tag{3.1}$$

$$f(i,j) < c(i,j), \quad \forall (i,j) \in E \tag{3.2}$$

L'équation (3.1) assure la conservation de flux, alors que (3.2) garantit le respect des capacités.

Dans le cas où les arcs sont pondérés, on utilise une fonction de coût supplémentaire noté u

$$u : E \to \mathbb{R}^+.$$

Problème du flot à coût maximal (ou minimal) : il consiste à trouver les flux qui maximisent (ou minimisent) le coût total, tout en respectant les contraintes de capacité et de conservation de flux. Dans le cas d'une maximisation, le problème est formulé comme suit

$$\max \sum_{(i,j)\in E} c(i,j) \cdot f(i,j)$$

sous contraintes (3.1) et (3.2).

Par contre, dans le cas d'une minimisation, la formulation devient

$$\max \sum_{(i,j)\in E} -c(i,j) \cdot f(i,j)$$

sous contraintes (3.1) et (3.2).

L'un des algorithmes les plus rapides pour la résolution de ce type de problème est le "*capacity scaling algorithm*" [2]. Mais, pour bien comprendre cet algorithme il est nécessaire de connaître la définition d'un graphe résiduel.

Graphe résiduel et capacité résiduelle :
Soient $G = (V, E, c)$ un graphe, et f^1 un flot dans ce graphe. Pour construire le graphe résiduel de G, on remplace chaque arc (i, j) de G par deux arcs (i, j) et (j, i). L'arc (i, j) aura une capacité résiduelle $r(i, j) = c(i, j) - f^1(i, j)$, et l'arc (j, i) aura une capacité résiduelle $r(j, i) = f^1(i, j)$. Le graphe résiduel contiendra uniquement les arcs qui ont des capacités résiduelles positives.

Capacity scaling algorithm :
L'idée de cet algorithme est d'augmenter le flot le long d'un chemin qui a une *capacité résiduelle suffisamment grande*, au lieu de l'augmenter sur le chemin qui a la plus grande

capacité résiduelle, [2].

Soient Δ un réel positif, et x un flot. On appelle Δ-*graphe résiduel*, tout graphe dans lequel les capacités des arcs sont chacune supérieure ou égale à Δ. Soit $G(x, \Delta)$ un Δ-*graphe résiduel*, (notons que $G(x, 1) = G(x)$ et que $G(x, \Delta)$ est un sous graphe de $G(x)$). On appelle *phase d'escalade*, une phase dans laquelle Δ garde la valeur qu'il avait à l'étape précédente de l'algorithme, alors qu'une phase avec une nouvelle valeur de Δ est appelée Δ-*phase d'escalade*. Dans une Δ-*phase d'escalade* chaque augmentation ajoute au moins Δ unités de flot. L'algorithme commence avec $\Delta = 2^{\lfloor \log(U) \rfloor}$, sachant que $U = \max_{(i,j)\in E} (i, j)$. Ensuite la valeur de Δ est divisée par deux à chaque *phase d'escalade*. L'algorithme s'arrête lorsque Δ devient égal à 1. Par conséquent, il contient $1 + \lfloor \log U \rfloor = O(\log U)$ *phases d'escalades*. En plus de cela, à la dernière *phase d'escalade*, on aura $\Delta = 1$, et donc $G(x, \Delta) = G(x)$. Cela montre que l'algorithme se termine avec un flot maximal. L'algorithme progresse comme suit.

1 : $x = 0$
2 : $\Delta = 2^{\lfloor \log U \rfloor}$
3 : Tant que $\Delta \geq 1$ faire
    3.a : Tant que $G(x, \Delta)$ contient un chemin allant de s à p faire
        - Identifier un chemin P dans $G(x, \Delta)$
        - $\delta = \min\{r(i, j) : (i, j) \in P\}$
        - Augmenter δ unités de flot le long de P et mettre à jour $G(x, \Delta)$

    Fin tant que
    3.b : $\Delta = \frac{\Delta}{2}$
Fin tant que

Ces outils de la théorie des graphes sont exploités différemment dans la résolution du problème de stockage de conteneurs.

Dans [75], Lee et Hsu ont transformé le problème de relocalisation de conteneurs en un problème de flot multiple dans un graphe orienté. Selon eux, le fait de repositionner, dans le bon ordre, les conteneurs qui sont destinés à un navire permet d'accélérer le chargement de ce dernier en minimisant (voir même en rendant nul) le nombre de remaniements qui sont prévus durant cette période. Ils ont considéré que ces manutentions sont effectuées par des grues de cour automatisées (RMGCs), et que chaque conteneur déplacé sera à nouveau stocké dans un emplacement qui est à l'intérieur de sa travée d'origine. Dans le graphe qu'ils ont proposé, chaque nœud représente un emplacement de stockage, les arcs correspondent à des opportunités de mouvements de conteneurs, et les flux dans le graphe représentent les mouvements des conteneurs dans l'espace temps. À partir de ce graphe, les auteurs ont proposé un modèle mathématique linéaire dans lequel tous les conteneurs considérés sont destinés à un seul navire et sont placés dans une même travée. En plus de cela, ils ont supposé que seuls des conteneurs qui ont les mêmes dimensions peuvent être superposés (dans chaque pile de la travée) et que l'ordre dans lequel ils seront placés sur le navire est connu bien avant le début du chargement (mais il n'était pas encore connu au moment de l'arrivée de ces conteneurs au port). Pour tester ce modèle, Lee et Hsu l'ont d'abord codé en C++ pour le résoudre avec CPLEX 9.0, mais ils ont constaté que

les durées d'exécution sont très longues. Ainsi, pour diminuer les temps de résolution, ils ont proposé un algorithme heuristique qui contient essentiellement deux étapes. Dans la première phase, l'algorithme essaie de trouver un ensemble de mouvements qui conduisent à une bonne configuration de la travée. Ensuite dans la deuxième étape, l'heuristique trie ces mouvements et élimine les cycles en ajoutant des mouvements supplémentaires. Lee et Hsu ont affirmé que cet algorithme heuristique est très rapide et ont suggéré d'étendre cette recherche en traitant le cas où plusieurs navires sont considérés.

Dans [111], Yu et Qi ont utilisé le *Capacity scaling algorithm* pour résoudre le problème de stockage de conteneurs importés (qui sont amenés au port par des navires), en suivant deux stratégies de stockage différentes. La première est non ségrégative (c'est à dire qu'elle n'interdit pas le stockage de conteneurs provenant de navires différents dans une même travée), contrairement à la deuxième qui est ségrégative et qui est myope car ne prenant en considération que les conteneurs qui sont arrivés dans la période de stockage courante (cette méthode est appelée le "*single-period segragation*"). Yu et Qi ont aussi proposé une troisième stratégie de stockage qu'ils ont appelée le "*multiple-period segregation*", dont la seule différence avec la deuxième stratégie est le fait qu'elle prend en considération les conteneurs qui vont arriver dans les futurs périodes de l'horizon de planification. Pour chacune de ces trois stratégies, Yu et Qi ont résolu le problème de stockage en deux étapes. Dans la première étape, ils déterminent le nombre de conteneurs affectés à chaque travée (sans préciser l'emplacement exact de chaque conteneur). Ensuite, dans la deuxième étape, ils résolvent le problème de relocalisation, en considérant une durée limite de manutention. L'objectif visé dans ces deux étapes est la minimisation des durées d'extraction des conteneurs lorsqu'ils sont réclamés par leurs propriétaires. Mais, vu que les auteurs ont supposé que les dates de réclamation des conteneurs sont complètement inconnues, alors ils ont exprimé la durée d'extraction d'un conteneur en fonction du nombre de conteneurs qui sont dans la même travée que lui. Ce qui signifie qu'ils n'ont pas cherché à déterminer les nombres de remaniements prévus lors des extractions. D'ailleurs, dans la phase de relocalisation, ils ne précisent pas les conteneurs qui sont déplacés d'une travée vers une autre, mais ils changent la répartition des conteneurs dans les travées (autrement dit, ils recalculent le nombre de conteneurs que doit contenir chaque travée). Pour la résolution de la première phase de stockage, Yu et Qi ont proposé trois modèles mathématiques, dont un pour chaque stratégie. Ils ont remarqué que les modèles correspondants aux deux premières stratégies ont des fonctions objectifs qui sont constituées de coûts convexes séparables, et ils ont donc transformé ces problèmes en des problèmes de flot minimal dans des graphes orientés dont les nœuds représentent des travées. Par la suite, ils ont résolu chacun de ces deux problèmes de flot en utilisant le *Capacity scaling algorithm*. En ce qui concerne la troisième stratégie, les auteurs ont proposé un algorithme de programmation dynamique pour sa résolution. Un seul modèle mathématique a été proposé pour le problème de relocalisation, indépendamment des trois différentes stratégies relatées dans les lignes précédentes. Cependant, contrairement au problème d'allocation (de la première phase), Yu et Hi n'ont pas résolu le problème de relocalisation avec une méthode exacte ; par contre, ils ont proposé un algorithme heuristique pour sa résolution. Ce dernier contient plusieurs itérations dans lesquelles sont effectuées des actions répétitives qui consistent à déplacer des conteneurs entre des travées différentes dans le but de réduire les durées d'extraction prévues. L'algorithme s'arrête lorsque la durée limite autorisée pour effectuer les déplacements est atteinte ou

bien lorsqu'il n'est plus possible de diminuer les durées d'extraction prévues. Les auteurs ont considéré que les manutentions sont effectuées par des grues automatisées (RMGCs), et que les opérations de relocalisation des conteneurs sont effectuées pendant la nuit. Après avoir effectué des simulations numériques, en considérant un horizon de planification long de trois mois (qui est divisé en des périodes qui sont chacune équivalente à un jour), les auteurs ont remarqué que le *multiple-period ségrégation* procure une durée moyenne de retrait inférieure à celle qui est obtenue en utilisant le *single-period segregation*. En plus de cela, ils ont aussi constaté que la méthode de *non ségrégation* est plus adaptée au problème, car elle fournit les durées d'extraction les plus faibles, et qu'elle diminue le risque de manque de place. En perspective, Yu et Qi ont suggéré d'étudier le cas du stockage où tous les conteneurs n'ont pas la même probabilité d'être retirés à n'importe quel moment.

3.3.6 Algorithmes génétiques

Les algorithmes génétiques sont créés en 1975 par le chercheur américain Jonh Henry Holland, qui a publié, dans [49], le premier modèle formel appelé the "*canonical genetic algorithm*". Ils font partis des algorithmes évolutionnaires, et sont basés sur les processus naturels de reproduction animale. En effet, ces processus sont mieux connus grâce aux progrès des recherches effectuées sur les cellules animales, qui ont révélé l'existence de deux phénomènes remarquables qui se produisent au cours de la reproduction animale, à savoir : le croisement et la mutation.

Le croisement, comme son nom l'indique, est le phénomène qui associe une partie d'un chromosome masculin à une autre partie d'un chromosome féminin, pour créer un nouveau chromosome. La figure suivante en est une illustration.

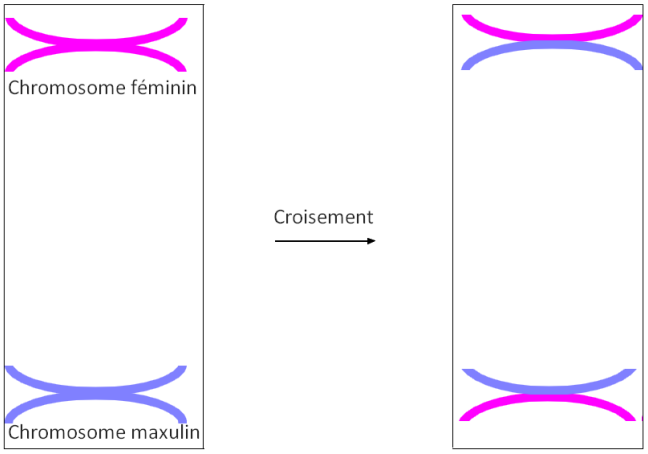


FIGURE 17 – Croisement cellulaire

La mutation est un phénomène naturel très rare qui n'est pas indispensable dans la reproduction animale. Comme son nom l'indique, elle modifie certaines informations contenues dans un chromosome (voir la figure 18), et par conséquent elle change certains caractères du nouvel individu créé. Dans la vie réelle elle peut avoir des conséquences néfastes telles que des malformations et des maladies génétiques. Mais, depuis quelques

temps, les chercheurs essaient d'exploiter les aspects positifs de ce phénomène, surtout dans le domaine de l'agriculture. D'ailleurs, ce phénomène a une utilité en optimisation, car elle peut éviter une convergence rapide vers un optimum local.

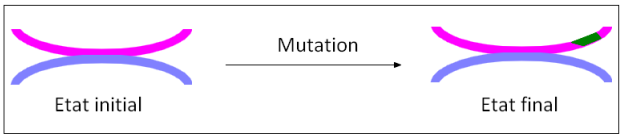


FIGURE 18 – Mutation cellulaire

L'implémentation d'un algorithme génétique requiert la détermination préalable de six paramètres qui sont détaillés ci-dessous.

1) **Un principe de codage**, autrement dit une manière de représenter un individu de la population étudiée. Les types de codage les plus fréquents dans la littérature sont : le codage binaire (qui est représenté par une succession de 0 et 1), le codage à caractères multiples, et le codage sous forme d'arbre (qui utilise une structure arborescente, à partir de laquelle pourront être issus un ou plusieurs fils). Il est capital de bien choisir un codage adéquat au problème étudié, car cela conditionne le succès de l'algorithme.

2) **Une manière de générer la population initiale**, qui doit permettre de créer des individus qui ne sont pas identiques. L'efficacité ou la non-efficacité de cette méthode se répercute dans les résultats finaux de l'algorithme, car une population initiale de très mauvaise qualité peut conduire à une convergence prématurée vers un optimum local, qui est très éloigné de l'optimum global. Il est donc très avantageux d'avoir une population initiale qui est bien répartie sur tout le domaine de définition du problème étudié.

3) **Une fonction d'évaluation**, qui permet de calculer la performance (appelée fitness) de chaque individu de la population. Elle sert aussi à connaître le meilleur individu (solution courante) d'une population, mais elle est souvent aussi utilisée pour sélectionner des individus à partir desquels la génération suivante sera créée.

4) **Un opérateur de croisement**, qui permet de créer des descendants de la population initiale, il permet aussi aux anciennes générations de transmettre leurs caractères aux nouvelles générations. Les méthodes de croisement les plus rencontrées dans la littérature sont : le croisement en un point, le croisement en deux points, la méthode du couper-et-joindre, le croisement uniforme, et le croisement avec trois parents.

- **Croisement en un point** : cette méthode choisit un point de section quelconque, ensuite elle intervertit les parties qui sont à sa droite dans les deux parents. La figure 19 en est une illustration.

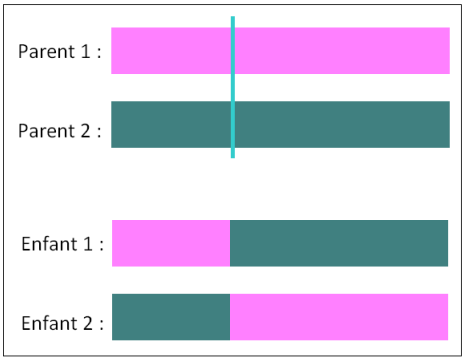


FIGURE 19 – Croisement en un point

- **Croisement en deux points** : elle est semblable au croisement en un point, mais deux points de section sont choisis au lieu d'un seul. Ensuite les parties qui sont entre ces deux sections sont échangées, comme le montre la figure 20.

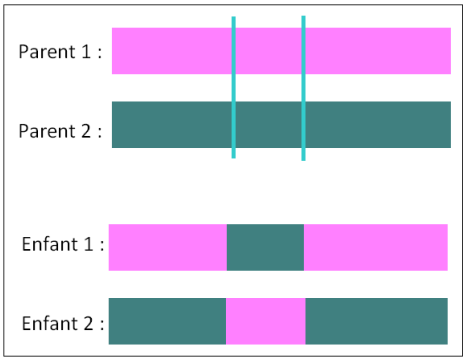


FIGURE 20 – Croisement en deux points

Une variante de cette méthode est le croisement partiel (ou "*partially mapped crossover*"), qui fut proposé par Goldberg et Lingle [45]. L'originalité de cette méthode est le fait qu'elle fournit une technique de correction des nouveaux chromosomes afin de remédier aux possibles violations de certaines contraintes. Elle progresse comme suit.

1 : Choisir aléatoirement deux points de section.
2 : Échanger les parties qui sont à leurs intérieurs, pour créer deux nouveaux enfants.
3 : Déterminer un processus de correction.
4 : Utiliser ce processus pour légaliser les enfants.

Le croisement partiel est à son tour amélioré par Davis [45], qui a proposé une autre version appelée "*order crossover*". Les principales étapes de cette méthode de croisement sont :

1 : Sélectionner une partie d'un parent.
2 : Créer un enfant vide, ensuite copier cette partie au bon endroit.
3 : Effacer du deuxième parent tous les symboles appartenant à la partie sélectionnée dans le premier parent. Les éléments restants sont ceux dont l'enfant a besoin.
4 : Remplir de gauche à droite les cases vides de l'enfant, en y plaçant les éléments restants du deuxième parent.

- **Méthode du couper-et-joindre** : c'est une variante du croisement en un point, qui change la longueur des chromosomes. Cela est dû au fait que différents points de section sont considérés dans les deux parents, par conséquent les enfants qui en résultent auront des tailles variées (voir la figure 21).

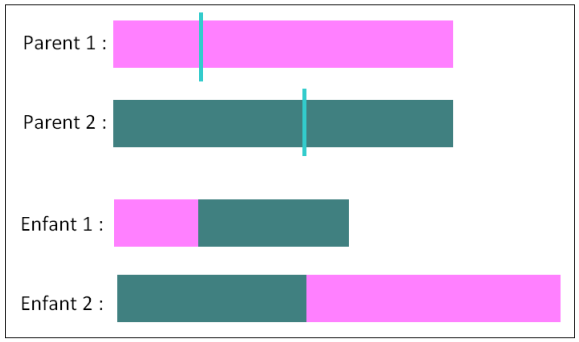


FIGURE 21 – Méthode du couper-et-joindre

- **Croisement uniforme** : cette méthode est créée par Syswerda, elle est spécifique aux chromosomes codés sous forme de chaînes de bits [45]. Dans un premier temps, elle génère aléatoirement un modèle sur lequel elle s'appuie pour échanger les gènes des parents. Ainsi, elle produit deux enfants qui ont les mêmes tailles que leurs parents.

Une amélioration du croisement uniforme appelée *croisement basé sur les positions* est proposée par Syswerda [45]. Celle-ci ajoute un processus de correction au croisement uniforme, son déroulement est presque semblable à celui du *order crossover* ; la seule différence se trouve au niveau de la première étape. L'algorithme du *croisement basé sur les positions* est le suivant.

1 : Choisir aléatoirement un ensemble de positions dans un parent.
2 : Créer un enfant vide et copier cette partie au bon endroit.
3 : Effacer du deuxième parent tous les symboles appartenant à la partie sélectionnée dans le premier parent. Les éléments restants sont ceux dont l'enfant a besoin.
4 : Remplir de gauche à droite les cases vides de l'enfant en y plaçant les éléments restants du deuxième parent.

Une légère modification du croisement basé sur les positions, appelée *croisement basé sur les ordres* est aussi proposée par Syswerda [45]. La seule différence est qu'avec cette méthode, lorsqu'on sélectionne des positions dans un des deux parents, les symboles qui sont dans les positions correspondantes dans l'autre parent conservent également leur ordre.

- **Croisement basé sur un cycle** : il est proposé par Oliver, Smith, et Holland. Comme pour le croisement basé sur les positions, il prend quelques symboles à partir de l'un des deux parents, ensuite, il prend le reste à partir de l'autre parent. La différence est que le choix des symboles ne se fait pas de façon aléatoire, seuls les symboles dont les positions correspondantes dans les deux parents forment un cycle peuvent être sélectionnés [45]. Les principales étapes de cette méthode de croisement sont les suivantes.

1 : Trouver un cycle qui est défini par les positions correspondantes des symboles dans les deux parents.
2 : Copier les symboles du cycle dans un enfant en respectant les positions correspondantes dans l'un des deux parents.
3 : Déterminer les symboles manquants dans l'enfant en enlevant de l'autre parent les symboles qui sont déjà utilisés.
4 : Remplir l'enfant avec les symboles restants.

- **Croisement avec trois parents** : cette méthode fait intervenir trois parents dans la création d'un enfant. Elle compare chaque bit du premier parent avec son correspondant dans le deuxième parent, s'ils sont pareils alors le bit est reporté dans l'enfant. Sinon, le bit du troisième parent est transmis à l'enfant, comme le montre l'exemple ci-dessous.

Parent 1 : 1 1 0 1 0 0 0 1 0
Parent 2 : 0 1 1 0 0 1 0 0 1
Parent 3 : 1 1 0 1 1 0 1 0 1
Enfant : 1 1 0 1 0 0 0 0 1

5) Le cinquième paramètre de l'algorithme génétique est l'**opérateur de mutation**. Ce dernier effectue une légère perturbation sur un chromosome en altérant un ou plusieurs de ses gènes. Il permet essentiellement de maintenir la diversité de la population, mais il peut parfois améliorer la valeur (performance) d'un chromosome. En plus de cela, il peut éviter une convergence prématurée de l'algorithme génétique. Les méthodes de mutation les plus connues dans la littérature sont : le basculement d'un bit, l'utilisation d'une borne, la méthode non-uniforme, la méthode uniforme, la méthode gaussienne, et la méthode d'échange.

- **Le basculement d'un bit** : il concerne uniquement le cas où les chromosomes sont codés en binaire, et consiste simplement à changer le contenu d'un gène (0 devient 1, et 1 devient 0).

- **L'utilisation d'une borne** : cette méthode consiste à remplacer un gène choisi aléatoirement, par sa borne supérieure ou bien inférieure.

- **La méthode non-uniforme** : avec cette méthode, la probabilité de mutation change au cours de l'algorithme génétique. Au début, elle reste faible pour garder la population

stagnante, mais vers la fin elle devient élevée afin d'améliorer la valeur des solutions.

- **La méthode uniforme** : elle remplace un gène par un nombre aléatoire choisi entre sa borne supérieure et sa borne inférieure.

- **La méthode gaussienne** : elle choisit un gène, et ajoute à son contenu une valeur choisie selon la loi gaussienne. Le nouveau gène obtenu est écarté s'il est supérieur à la borne supérieure ou bien s'il est inférieur à la borne inférieure. Cette méthode ainsi que les trois précédentes sont spécifiques aux chromosomes contenant des nombres entiers ou bien des nombres réels.

- **La méthode d'échange** : elle consiste simplement à choisir arbitrairement deux gènes et à intervertir leurs contenus.

6) L'algorithme génétique fait aussi intervenir des **paramètres numériques**, qui sont : la taille maximale d'une population (NI_{Max}), le nombre de générations successives à créer (NG_{Max}), la probabilité de croisement (pc), et la probabilité de mutation (pm).

Une description de la succession des étapes de l'algorithme génétique est la suivante.

1 : Créer une population initiale.
2 : Évaluer chaque individu.
3 : Initialiser le compteur du nombre de générations, $NG = 1$.
4 : Tant que $NG < NG_{Max}$ faire :
  4.a : Sélectionner une partie de la population courante.
  4.b : Initialiser une *nouvelle génération* qui contient les individus sélectionnés.
  4.c : Tant que la nouvelle génération ne contient pas NI_{max} éléments, faire :
    - Choisir dans la population sélectionnée deux parents (P_1 et P_2).
    - Choisir aléatoirement entre 0 et 1 un réel noté c.
    - Si $c < pc$, alors faire un croisement entre P_1 et P_2 pour créer deux enfants.
    - Ajouter dans la *nouvelle génération* le meilleur enfant.

    Fin tant que.
  4.d : Choisir aléatoirement entre 0 et 1 un réel noté m.
  4.e : Si $m < pm$, alors faire une mutation sur un individu choisi aléatoirement dans *la nouvelle génération.*
  4.f : Evaluer la *nouvelle génération.*
  4.g : Incrémenter le compteur, $NG = NG + 1$.
Fin tant que.

La sélection des individus à partir desquels on crée une nouvelle génération peut être faite de plusieurs façons. Les méthodes de sélection les plus connues sont : la méthode élitiste, la roue de la fortune, la sélection par rang, la sélection par tournoi, et la sélection uniforme.

- **Méthode élitiste** : elle sélectionne les meilleurs individus en se basant sur leurs fitness. L'avantage de cette méthode est le fait qu'elle permet de ne pas perdre les meilleurs individus au cours des générations, mais cela augmente aussi le risque d'une convergence prématurée vers un optimum local.

- **Méthode de la roue de la fortune** : c'est la méthode de sélection la plus fré-

quente dans la littérature. Avec elle, chaque individu a la possibilité d'être choisi selon une probabilité qui dépend de sa performance. Ainsi, les individus qui sont plus adaptés au problème ont plus de chance d'être sélectionnés. Comme son nom l'indique, cette méthode répartit les individus de la population sur une roue mesurant 360°. Chacun d'eux a une proportion qui est équivalente à sa fitness. Autrement dit, chaque élément e de la population Pop, qui a une une fitness f_e, a une portion mesurant $\frac{360 \times f_e}{\sum_{v \in pop} f_v}$ degrés (voir la figure 22). De ce fait, pour sélectionner un individu, il suffit de faire tourner la roue et de choisir l'individu sur lequel pointe le curseur. Le principal inconvénient de cette méthode est qu'il peut arriver que la majeur partie (voire même la totalité, dans le pire des cas) des individus sélectionnés aient des fitness médiocres, ce qui revient à dire que les individus de la future génération seront créés à partir de parents qui ont des faibles performances. Cela est contre l'idéologie des algorithmes génétiques qui visent à obtenir continuellement, au cours des générations, des individus qui ont des performances meilleures que celles de leurs ascendants. Un autre risque à encourir avec cette méthode est le fait que si un individu a une fitness largement plus intéressante que celles des autres, alors il est possible qu'au bout d'un certain nombre de générations, d'avoir une population qui n'est composée que de copies de cet individu. Cela entraîne aussi une convergence prématurée, l'algorithme restera donc bloqué sur un optimum local, et l'optimum global ne sera jamais atteint. Une méthode pour éviter ce problème est d'imposer que tous les individus choisis lors d'une sélection soient différents ; autrement dit, même si un individu est pointé plusieurs fois par le curseur, on ne l'ajoute qu'une seule fois dans la population sélectionnée.

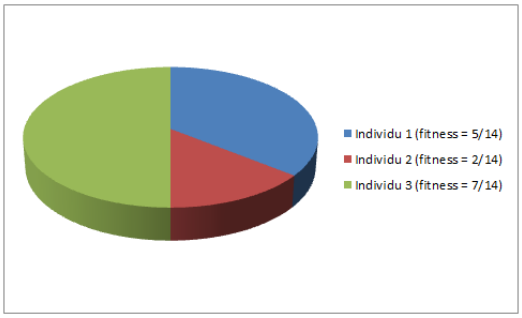


FIGURE 22 – Roue de la fortune

- **Sélection par rang** : elle consiste à trier les individus d'une population suivant leurs fitness, ensuite les numéroter par ordre décroissant. Ce qui signifie que l'individu qui a la plus petite fitness aura le numéro 1, et ainsi de suite. Cette méthode fonctionne presque de la même façon que celle de la roue de la fortune, avec la seule différence que la proportion de la roue accordée à chaque individu dépend de son rang et non pas de sa fitness. Avec cette méthode, chaque individu a la possibilité d'être sélectionné mais la convergence vers la bonne solution est plus lente qu'avec la méthode de la roue de la fortune.

- **Sélection par tournoi** : elle consiste à effectuer des tirages successifs avec remise de deux individus dans la population. À chaque tirage, une compétition permet de départager les deux candidats. Cette dernière se fait indépendamment des autres individus de la population. Autrement dit, la probabilité de victoire de chaque individu ne dépend que

de ses performances et de celles de son adversaire. Cependant, la probabilité de victoire de l'individu qui a la plus grande fitness est préalablement fixée entre 0.7 et 1, car elle est considérée comme un paramètre de l'algorithme génétique. Avec cette méthode, on peut varier la pression de la sélection en augmentant ou bien en diminuant la probabilité de victoire de l'individu le plus intéressant.

- **Sélection uniforme** : elle se fait de façon aléatoire, uniformément, sans l'influence de la fitness. Avec cette méthode, il y a équiprobabilité de sélection entre les individus de la population.

Les algorithmes génétiques sont utilisés dans la résolution de divers problèmes. Dans [57], Kozan et Preston s'en sont servi pour résoudre le problème de stockage de conteneurs. Ils ont considéré un terminal multi-modal, dans lequel les conteneurs transitent entre camions, trains, et navires. Leur étude englobe aussi bien les conteneurs entrants (qui sont en importation) que les conteneurs sortants (qui sont en exportation) et ceux qui sont en transition (c'est-à-dire qui sont déchargés de certains navires et qui seront chargés sur d'autres navires). Dans un premier temps, les auteurs ont proposé un modèle mathématique de stockage de conteneurs qui a pour objectif de minimiser les durées de séjour des navires. Cependant, ils n'ont pas tenu compte, dans ce modèle qui est sous forme de programme linéaire mixte en nombres entiers, de la diversité des tailles entre les conteneurs, ni des contraintes gravitationnelles. Pour la résolution numérique du problème, les auteurs ont proposé un algorithme génétique qu'ils ont codé en Visual Basic en utilisant Microsoft Access. Dans leur algorithme, ils ont représenté chaque solution sous la forme d'une chaîne de nombres entiers dont la taille est égale au nombre de conteneurs à stocker. Chaque conteneur doit apparaître une seule fois dans chaque chromosome (solution). Cependant, dans le cas où le nombre de conteneurs n'est pas divisible par le nombre de véhicules internes de transport, les auteurs ajoutent à chaque chromosome des conteneurs fictifs de sorte que les nombres de conteneurs affectés aux véhicules soient égaux. Ainsi, chaque solution est composée de plusieurs segments qui contiennent chacun des conteneurs qui sont affectés à un même véhicule de transfert (qui peut être un cavalier, ou un camion, ou un véhicule auto-guidé). En plus de cela, chaque véhicule transportera ses conteneurs suivant l'ordre indiqué dans le chromosome. Dans chacune de leurs simulations, Kozan et Preston ont travaillé avec 60 générations ayant chacune 30 individus. Ils ont testé deux méthodes de croisement : le croisement en un point et le croisement en deux points. Mais ils ont remarqué que le croisement en deux points nécessite beaucoup trop de temps même s'il donne des résultats légèrement meilleurs que ceux du croisement en un point. La méthode de sélection qu'ils ont utilisée est une version légèrement modifiée de la roulette de la chance, dans laquelle les deux meilleurs individus sont immédiatement sélectionnés. Quant à la mutation, elle est faite en échangeant la position de deux conteneurs dans un chromosome. La stratégie proposée dans ce papier est non ségrégative, et a pour objectif de stocker les conteneurs dans les emplacements les plus proches des navires, tout en minimisant les durées des manœuvres et des transports. Selon les auteurs, cette stratégie est plus efficace que le stockage aléatoire. Mais ils ont également constaté que, même avec cette méthode, les temps de transfert augmentent considérablement lorsque le nombre d'équipements de manutention et de transport diminue. En plus de cela, ils ont remarqué que le nombre maximal de conteneurs autorisés dans une pile a une influence sur la durée totale des transports, et ont proposé de limiter

ce nombre à trois. L'algorithme génétique proposé dans cet article semble prometteur, mais, néanmoins, les auteurs ne l'ont pas comparé avec une méthode exacte (sur des instances de petites tailles, pour savoir s'il procure des solutions proches des optimums globaux), ni avec d'autres méta-heuristiques.

Une étude plus étendue que cette précédente est proposée dans [58] par Kozan et Preston, où sont traités de façon cyclique le problème de transfert de conteneurs (CTM) et celui du stockage de conteneurs (CLM). Chacun de ces deux sous-problèmes est résolu séparément, mais les résultats de l'un sont les données d'entrée de l'autre et vice versa. Les auteurs se sont particulièrement intéressés au stockage des conteneurs qui sont en exportation, et aussi à l'ordonnancement des trajets à effectuer pour transporter ces conteneurs de la cour de stockage vers les quais. Ils ont considéré une stratégie de ségrégation qui ne mélange pas, dans une zone, les conteneurs qui seront chargés sur des navires et ceux qui partiront par voie routière ou ferrée. L'objectif visé par les auteurs, dans la résolution du CTM, est de trouver le plan de transfert de conteneurs qui minimise les durées de séjour des navires, autrement dit, les durées des transports entre la cour de stockage et les quais. Tandis que celui recherché dans le CLM est de trouver le plan de stockage optimal qui minimise les durées de manutention des conteneurs. Selon Kozan et Preston, l'utilité de combiner ces deux problèmes dans un seul modèle se justifie par le fait qu'il soit vain d'optimiser l'un alors que l'autre est loin d'être optimal. De ce fait, ils ont proposé un modèle mathématique (modèle intégré) qui est commun au CLM et au CTM, dans lequel les variables de décision de l'un sont des données d'entrées de l'autre et vice versa. Ce modèle est par la suite résolu avec deux techniques itératives différentes. Ces deux techniques utilisent chacune deux algorithmes génétiques : l'un pour la résolution du CLM et l'autre pour la résolution du CTM. Cependant, la différence entre ces deux techniques est le fait que : dans celle que les auteurs ont nommé *non croissant*, le nombre de générations est constant à chaque itération, alors que dans celle qu'ils ont appelée *croissant* le nombre de générations augmente au cours des itérations. Les paramètres (probabilité de croisement, probabilité de mutation, nombre de générations, et nombre de chromosomes) utilisés dans ces algorithmes génétiques ont des valeurs différentes mais la méthode de la roulette de la chance est communément utilisée pour effectuer des sélections. Les auteurs ont implémenté ces algorithmes en utilisant le langage C++, en plus de cela, ils ont proposé une méthode hybride dans laquelle un algorithme génétique est utilisé pour résoudre le CTM alors qu'un algorithme tabou est utilisé pour résoudre le CLM. Après avoir effectué des comparaisons entre ces différents algorithmes, Kozan et Preston ont constaté que l'algorithme itératif intégré procure généralement des résultats qui sont meilleurs que ceux obtenus en résolvant séparément le CLM et le CTM. En plus de cela, ils ont également remarqué que l'hybridation qu'ils ont proposée n'améliore pas les résultats.

Dans [7], Bruzzon et Signorile ont proposé deux algorithmes génétiques ; l'un pour résoudre le problème d'allocation de postes à quai, et l'autre pour résoudre le problème de stockage de conteneurs. Ils ont réuni ces deux algorithmes dans un simulateur qu'ils ont codé en langage C. Selon eux, ces deux problèmes sont liés car ils ont des paramètres en commun tels que : les dates d'arrivées prévues des navires, les types de chargement, et les emplacements des conteneurs dans la cour de stockage. Dans le premier algorithme, la fonction d'évaluation est une combinaison entre les dates d'arrivées prévues des navires et les quantités de conteneurs qu'ils transportent ; chaque navire est désigné par un vecteur

contenant ces deux types d'information. Les chromosomes de cet algorithme sont représentés par des chaînes de nombres réels. Cependant, les auteurs ont utilisé une méthode de croisement originale qu'ils ont nommée *order-based crossover*. Cette méthode sélectionne un ensemble de navires, ensuite elle intervertit leurs ordres de succession dans les deux parents pour créer deux nouveaux enfants. Mais seuls les nouveaux chromosomes qui ont des fitness (performances) supérieures à une valeur seuil sont ajoutés dans la nouvelle génération ; cela permet d'accélérer la convergence, d'après Bruzzon et Signorile. Quant à la mutation, elle est faite de façon classique en intervertissant les quais alloués à deux navires quelconques. Dans les deux algorithmes génétiques, les auteurs ont utilisé la méthode de sélection élitiste. À la fin du premier algorithme, les résultats obtenus sont utilisés dans le deuxième algorithme génétique pour trouver les meilleurs emplacements à allouer aux conteneurs qui vont être chargés sur les navires. Mais, vu que les auteurs ont opté pour la stratégie de stockage par groupe, l'objectif du deuxième algorithme est surtout de déterminer les groupes de piles qui sont réservés aux conteneurs de chaque navire (des groupes de piles réservés à un même navire peuvent être éloignés les uns des autres). Cet algorithme traite les navires individuellement, et vise à minimiser la distance totale parcourue entre la cour de stockage et chaque navire. Ainsi, les emplacements des navires sur les quais et les zones de stockage dans la cour sont représentés dans chaque chromosome. Contrairement à ceux du premier algorithme, les chromosomes du deuxième algorithme génétique sont des chaînes de bits en nombre binaire, chacun d'entre eux est divisé en des sous-chaînes contenant les coordonnées des emplacements des conteneurs et des navires. Quant aux opérations de mutation et de croisement, elles sont faites respectivement en changeant aléatoirement la valeur d'un bit et en suivant la règle du croisement basé sur les positions. Cependant, pour assurer la diversification des individus de chaque génération, les auteurs ont imposé que les chromosomes soient deux à deux distincts. En plus de cela, ils ont utilisé une fonction de pénalité pour éliminer les solutions non-faisables et celles qui n'ont pas une fitness supérieure au seuil. Dans leur simulateur, Bruzzone et Signorile ont modélisé les camions qui transportent les conteneurs dans la cour, de même que les grues de quai et les mouvements des conteneurs. Ils n'ont pas comparé leurs algorithmes génétiques à une méthode exacte, mais ils ont comparé le deuxième algorithme (c'est-à-dire l'algorithme génétique qui résout le problème d'allocation d'espaces de stockage) à un algorithme de recuit simulé qui utilise les résultats du premier algorithme (c'est-à-dire l'algorithme génétique qui résout le problème d'allocation de postes à quai). Selon eux, les résultats de l'algorithme génétique sont meilleurs que ceux du recuit simulé. D'autres comparaisons réalisées sur quatre différents cas ont été soulignées, dans cet article, pour justifier l'utilité de combiner le problème d'allocation de postes à quai et le problème de stockage de conteneurs. Dans le premier cas, la méthode du Premier-arrivé/Premier-servi est utilisée ; ce qui signifie qu'aucun des deux algorithmes génétiques précédents n'est utilisé et que les groupes de conteneurs (appartenant au même navire) sont placés selon leur ordre d'arrivée. Le deuxième et le troisième cas contiennent chacun respectivement l'un des deux algorithmes génétiques, à savoir : celui qui traite l'allocation des postes à quai et celui qui résout le problème de stockage de conteneurs. Quant au quatrième cas, il contient les deux algorithmes génétiques. Après avoir testé ces différentes techniques, les auteurs ont constaté que la méthode du Premier-arrivé/Premier-servi est moins performante que les autres. Cependant, la deuxième méthode fournit des résultats meilleurs que ceux de la troisième ; mais elle est moins performante que la quatrième méthode, qui

est la meilleure des quatre. L'approche proposée dans cet article est très pertinente, mais malheureusement les auteurs n'ont pas précisé les valeurs des paramètres utilisés dans leurs algorithmes génétiques ; néanmoins, ils ont souligné qu'ils envisagent d'analyser ultérieurement la sensibilité de ces paramètres.

3.3.7 Algorithme de recherche d'une harmonie

Cet algorithme, qui est inspiré des processus d'improvisation de musiciens, est initialement développé, dans [48], par Geem et al. En effet, ils ont remarqué que le principal objectif d'un musicien est de trouver une parfaite harmonie de sons. Ensuite, ils ont fait l'analogie entre l'harmonie musicale et le processus d'optimisation qui vise à atteindre l'optimalité. D'un côté, l'harmonie parfaite est déterminée par la qualité du son ; ce qui pousse le musicien à chercher davantage la meilleure harmonie. D'un autre coté, une solution optimale d'un problème d'optimisation doit être la meilleure solution qui est favorable aux objectifs et qui satisfait toutes les contraintes du problème. Ces deux processus ont un point commun qui consiste à atteindre le meilleur (l'optimalité).

Lorsqu'un musicien improvise, il a trois choix : soit il joue une pièce de musique populaire qu'il connaît, ou bien il le modifie légèrement, ou bien il compose une nouveauté. Geem et al. ont donc formalisé ces trois options en des processus d'optimisation quantitatifs [48], qui sont : l'utilisation d'une mémoire d'harmonies, l'ajustement des fréquences, et l'aléatoire (randomisation).

L'utilisation d'une *mémoire d'harmonies* est importante, car elle est similaire au fait de sélectionner les meilleurs individus d'une population dans un algorithme génétique. Elle assure que la meilleure harmonie de l'itération courante fera partie de la mémoire d'harmonies de la prochaine itération. Dans le but d'utiliser la mémoire efficacement, un paramètre $r_{accept} \in [0, 1]$, appelé *taux d'acceptation dans la mémoire*, est utilisé dans l'algorithme. Si ce taux est trop bas, alors peu d'harmonies seront sélectionnées parmi les meilleurs à chaque itération ; par conséquent, la convergence de l'algorithme sera trop lente. Par contre, si ce taux est extrêmement élevé (proche de 1), alors presque toutes les harmonies seront sélectionnées. De ce fait, beaucoup d'autres harmonies ne seront pas explorées, et par conséquent, les solutions seront potentiellement mauvaises. Ainsi, par commodité, ce taux est généralement choisi entre $0,7$ et $0,95$.

En ce qui concerne l'ajustement des fréquences, il se fait en utilisant deux paramètres : une bande passante (notée b_{range}) et un taux d'ajustement de fréquence (dénommé r_{pa}). Bien qu'en musique, l'ajustement des fréquences signifie leurs changements, cependant, dans *l'algorithme de recherche d'une harmonie*, il correspond à la création d'une solution légèrement différente [48]. En théorie, l'ajustement des fréquences peut être fait linéairement ou non-linéairement, mais en pratique il est fait linéairement en utilisant l'équation suivante

$$x_{new} = x_{old} + b_{range} \times \epsilon$$

où x_{old} est une solution appartenant à la mémoire, et x_{new} est la nouvelle solution obtenue après l'ajustement. Cela produit essentiellement une nouvelle solution qui est proche de la solution précédente, en variant légèrement la fréquence avec une petite quantité choisie aléatoirement [48, 71]. Ici, ϵ est un nombre aléatoire compris entre -1 et 1. L'ajustement de son est similaire à l'opérateur de mutation dans un algorithme génétique. Pour contrôler le

degré d'ajustement, on utilise un *taux d'ajustement* noté r_{pa}. Un faible taux d'ajustement, associé à une restreinte bande passante, peut aussi ralentir la convergence de l'algorithme, en réduisant le domaine d'exploration à un petit sous-espace de l'espace total de recherche. Par contre, un taux d'ajustement très élevé, associé à une large bande passante, peut causer la dispersion des solutions autour de potentielles solutions optimales comme dans une recherche aléatoire. Ainsi, le taux d'ajustement (r_{pa}) est généralement choisi entre 0.1 et 0.5.

Le troisième composant de l'algorithme est la randomisation, qui sert à augmenter la densité des solutions. Bien vrai que l'ajustement de fréquence a un rôle similaire, cependant il est limité à l'ajustement d'une certaine fréquence locale, et par conséquent, il correspond à une recherche locale. L'utilisation de la randomisation peut conduire le système à explorer davantage diverses solutions variées, de sorte qu'il trouve l'optimum global.

L'*algorithme de recherche d'une harmonie* progresse comme suit

1 : Définir la fonction objectif.
2 : Générer des harmonies initiales (des tableaux de nombres entiers).
3 : Définir le taux d'ajustement de fréquence (r_{pa}), ainsi que les fréquences limites et la bande passante.
4 : Définir le taux d'acceptation dans la mémoire (r_{accept}).
5 : Poser $t = 1$.
6 : Tant que ($t <$ Nombre_maximal_d'itérations) faire
- 6.a : Générer de nouvelles harmonies en n'acceptant que les meilleures d'entre elles.
- 6.b : Ajuster les fréquences pour obtenir de nouvelles harmonies (solutions).
- 6.c : Choisir aléatoirement un nombre *rand*, compris entre 0 et 1.
- 6.d : Si ($rand > r_{accept}$) alors
  - Choisir aléatoirement une harmonie parmi celles qui existent déjà.
- 6.e : Sinon
  - Si ($rand > r_{pa}$)
    - ◇ Ajuster aléatoirement les fréquences sans dépasser les limites.
  - Sinon
    - ◇ Générer de nouvelles harmonies via la randomisation.
- 6.f : Accepter les nouvelles harmonies (solutions) si elles sont meilleures.
- 6.g : Incrémenter le nombre d'itérations, ($t = t + 1$).
- 6.h : Trouver les meilleures solutions de l'itération courante.

Fin tant que.

L'algorithme de recherche d'une harmonie est utilisé pour résoudre différents problèmes concrets. Dans [1], Ayachi et al. ont proposé une version de cet algorithme pour la résolution du problème de stockage de conteneurs dans un terminal portuaire. Ils se sont

particulièrement intéressés au stockage de conteneurs qui sont destinés à l'exportation (autrement dit, qui seront chargés sur des navires). Cependant, leur principal objectif était d'affecter chaque conteneur à un emplacement de stockage précis de telle sorte que le nombre total de remaniements prévu lors des chargements des navires soit minimal. Ils ont considéré différents types de conteneur (réfrigérés, vides, citernes, etc.). Cependant, ils ont adopté une stratégie de stockage dispersé qui permet d'affecter différents types de conteneur à une même travée tout en respectant les contraintes liées à la nature de ces conteneurs (par exemple : affecter les conteneurs réfrigérés dans des blocs équipés, etc.). Néanmoins, les auteurs n'ont pas précisé les équipements de manutention considérés, en plus de cela, ils ont supposé que tous les conteneurs ont les mêmes dimensions. Dans leurs simulations numériques, ils ont fixé les valeurs des paramètres : le nombre d'itérations est égal à 20, la mémoire d'harmonies a une capacité de 30 solutions, le taux d'ajustement de fréquence est égal à 0.1, et le taux d'acceptation dans la mémoire vaut 0.95. Après avoir effectué des tests de comparaison, Ayachi et al. ont conclu que l'algorithme de recherche d'une harmonie donne des résultats qui sont meilleurs que ceux de l'algorithme génétique et ceux de la méthode "*last-in/first-out*".

3.3.8 Algorithme de colonie de fourmis

L'algorithme de colonie de fourmis est basé sur le comportement de ces animaux lorsqu'ils collectent de la nourriture. En effet, des études concernant ces bestiaux ont révélé qu'ils sont capables de trouver le plus court chemin entre leur fourmilière et un endroit où se trouve de la nourriture. Les fourmis communiquent entre eux de façon indirecte grâce à une substance, appelée phéromone, qu'ils sécrètent continuellement et déposent le long des chemins qu'ils parcourent. Le but de cette action est de marquer les chemins qui sont parcourus afin de les indiquer aux autres fourmis. Cependant, puisque la phéromone est une substance qui s'évapore au cours du temps, sa quantité est donc plus élevée sur le chemin le plus fréquenté. Ainsi, vu que les fourmis sont non seulement capables de détecter la présence de phéromone mais aussi de mesurer son ampleur, ils vont naturellement privilégier le chemin qui en a plus.

Les travaux de Deneubourg et al. [34, 47] ont beaucoup contribué à l'élucidation du comportement des fourmis naturels. Deneubourg et al. ont relié une fourmilière et une source de nourriture par deux branches, pour étudier expérimentalement la relation entre les quantités de phéromone déposées et les comportements des fourmis. Ils ont effectué plusieurs expériences en faisant varier la différence entre les longueurs des deux branches. Ainsi, ils ont remarqué que les fourmis commencent toujours par choisir librement leurs chemins, mais dans la plupart des expériences, ils finissent tous par utiliser la branche la plus courte.

Ce résultat peut être expliqué par le fait qu'au début d'une expérience, il n'y a pas de phéromone sur les branches. Par conséquent, les fourmis n'ont pas de préférence. Ainsi, ils choisissent avec la même probabilité l'une ou l'autre des deux branches. De ce fait, il peut arriver que la moitié des fourmis choisisse une branche et que l'autre moitié choisisse l'autre branche ; mais le hasard peut aussi favoriser l'une des deux. Cependant, puisque l'une d'entre elles est plus courte que l'autre, les fourmis qui l'ont choisie seront les premiers à atteindre la source de nourriture et à entamer le chemin de retour. Ainsi, au moment de choisir le chemin de retour, la phéromone qu'il y a sur la plus courte

branche va les inciter à la choisir et y déposer encore plus de phéromone. Par conséquent, la phéromone va s'accumuler plus rapidement sur cette dernière, qui sera probablement utilisée par la majorité des fourmis.

Dans le cas général, les différents chemins possibles sont représentés par un graphe, dans lequel chaque fourmi se déplace étape par étape, en choisissant à chaque fois un arc à parcourir.

Le premier algorithme de colonie de fourmis, nommé *Ant System* (AS), fut créé par Dorigo [35, 36], vers les années 1992. Par la suite, des améliorations ont été effectuées, et d'autres variantes ont vu le jour. Les plus connues d'entre elles sont : la version nommée *rank-based* (AS_{rank}), le *MAX-MIN Ant System* (MMAS), et le *Ant Colony System* (ACS).

L'algorithme de colonie de fourmis peut être décrit comme suit :

1 : Initialisation du phéromone.
2 : Chaque fourmi construit une solution.
3 : Évaluation des solutions.
4 : Initialisation du nombre de voyages par fourmi, $NT = 1$.
5 : Tant que ($NT <$ Nombre_Maximal_D'itérations) faire :
    5.a : Mettre à jour les traces de phéromone.
    5.b : Chaque fourmi construit une nouvelle solution.
    5.c : Évaluer les solutions.
    5.d : $NT = NT + 1$.
Fin tant que.

Lorsqu'une fourmi construit une solution, elle commence aléatoirement par parcourir un arc. Ensuite, elle choisit les prochains arcs, un par un, avec la probabilité

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \quad \forall\, j \in N_i^k$$

où η_{ij} est à priori une information heuristique relative au problème étudié (généralement, on a $\eta_{ij} = \frac{1}{d_{ij}}$, où d_{ij} est la longueur de l'arc ij). α et β sont deux paramètres qui déterminent respectivement l'influence des traces de phéromone et celle de l'information heuristique, tandis que N_i^k est l'ensemble des arcs qui ne sont pas encore parcourus par la fourmi k et qui sont adjacents au sommet i.

La mise à jour des traces de phéromone assure deux fonctions : l'évaporation et l'augmentation. A chaque itération de l'algorithme, l'évaporation est effectuée sur toutes les traces de phéromone, en les multipliant par $(1 - \rho)$, (où $0 < \rho \leq 1$ est le taux d'évaporation). Par contre, l'ajout se fait différemment selon la version considérée de l'algorithme. En effet, avec l'algorithme AS, la phéromone est augmentée sur tous les chemins parcourus, à chaque itération, en utilisant la formule suivante :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t), \quad \forall (i,j) \tag{3.3}$$

où $\Delta\tau_{ij}^{k}(t)$ est la quantité de phéromone déposée par la fourmi k sur l'arc ij, elle est définie comme suit :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & \text{si l'arc } ij \text{ est parcouru par la fourmi } k \\ 0 & \text{autrement} \end{cases}$$

où $L^{k}(t)$ est la longueur du chemin parcouru par la $k^{\text{ème}}$ fourmi. Ainsi, plus un chemin est court, plus sa quantité de phéromone sera grande. En général, les arcs qui sont parcourus par plusieurs fourmis et qui appartiennent à des courts chemins vont recevoir plus de phéromone, et donc auront plus de chance d'être choisis dans les futures itérations de l'algorithme.

Pour améliorer les performances de l'algorithme AS, une *stratégie élitiste* fut proposée dans [35, 36] par Dorigo et al. Elle consiste à ajouter, à chaque itération, un surplus de phéromone sur le meilleur chemin (nommé T^{mg}, où mg signifie meilleur global) trouvé. Ainsi, les arcs appartenant à T^{mg} recevront la quantité additionnelle de phéromone suivante

$$\Delta\tau_{ij}^{mg}(t) = \begin{cases} e/L^{mg}(t) & \text{si l'arc } ij \in T^{mg} \\ 0 & \text{autrement} \end{cases}$$

où L^{mg} est la longueur du chemin T^{mg}, et e est un nombre entier positif.

La deuxième amélioration de l'algorithme AS est le (AS_{rank}) [10], qui est une extension de la stratégie élitiste. En effet, avec l'algorithme du AS_{rank}, les fourmis sont triées suivant l'ordre décroissant de la longueur des chemins qu'ils ont construits, à la fin de chaque itération. Ensuite, seules les $(w-1)$ fourmis et la meilleure-globale seront autorisées à ajouter de la phéromone, proportionnellement à leurs performances. La quantité de phéromone déposée par la $r^{\text{ème}}$ fourmi est égale à $\max\{0, w-r\}$, tandis que celle déposée par la meilleure-globale est égale à w. Ainsi l'équation (3.3) devient

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{r=1}^{w-1}(w-r)\cdot\Delta\tau_{ij}^{r}(t) + w\cdot\Delta\tau_{ij}^{mg} \tag{3.4}$$

où $\Delta\tau_{ij}^{r}(t) = 1/L^{r}(t)$ et $\Delta\tau_{ij}^{mg}(t) = 1/L^{mg}$.

Quant à l'algorithme ACS [38, 37, 44], il améliore l'algorithme AS en suivant deux mécanismes. Le premier consiste à ajouter de la phéromone uniquement sur les arcs appartenant à la meilleure solution (qui peut être celle de l'itération courante, ou bien la meilleure solution globale). Le deuxième mécanisme concerne le choix des arcs. En effet, avec l'algorithme ACS, il se fait suivant une règle nommée "*pseudo-random proportional*", qui, suivant une probabilité q_0, choisit le sommet j qui maximise le produit $\tau_{ij}(t)\cdot\eta_{ij}^{\beta}$, tandis qu'avec une probabilité $1-q_0$ elle procède comme dans l'algorithme AS. Ainsi, si $q_0 = 0$, la méthode du "*pseudo-random proportional*" fonctionne pareillement que celle de l'algorithme AS. Une autre différence de l'algorithme ACS par rapport au AS est le fait que l'évaporation des traces de phéromone est faite durant la construction des solutions. Cela est réalisée afin de diminuer la probabilité qu'un même chemin soit suivi par toutes les fourmis (en d'autres termes, ce procédé favorise l'exploration en relativisant

les effets des deux modifications citées ci-dessus, qui favorisent fortement l'exploitation des connaissances acquises sur le problème étudié). La dernière différence est le fait que l'algorithme ACS utilise un algorithme de recherche locale pour améliorer ses solutions en les transformant en des optimums locaux.

La troisième version d'algorithme de colonie de fourmis, nommée *MMAS* [100, 99, 101], introduit une borne supérieure et une borne inférieure pour les valeurs des traces de phéromone, ainsi qu'une méthode d'initialisation de la phéromone qui est différente de celle utilisée dans les précédentes versions. En pratique, avec l'algorithme *MMAS*, les valeurs des traces de phéromone sont comprises dans l'intervalle $[\tau_{min}, \tau_{max}]$, ce qui signifie que $\tau_{min} \leq \tau_{ij} \leq \tau_{max}, \ \forall \ \tau_{ij}$. En plus de cela, les traces de phéromone sont initialisées à la valeur maximale, ce qui favorise une large exploration de l'espace de recherche, au début de l'algorithme. Cependant, la mise à jour de la phéromone est faite comme dans la version ACS, en ajoutant de la phéromone uniquement sur les arcs qui appartiennent à la meilleure solution (qui peut être celle de l'itération courante, ou bien celle trouvée au niveau global). Une autre similarité de l'algorithme *MMAS* par rapport à ACS est le fait qu'il fait lui aussi appel à des algorithmes de recherche locale pour transformer ses solutions en des optimums locaux.

L'algorithme de colonie de fourmis a été initialement créé pour résoudre le problème du voyageur de commerce ; mais de nos jours, il est utilisé dans la résolution de divers problèmes.

Dans [12], Yalaoui et al. ont proposé un algorithme de colonie de fourmis, nommé MOACO, pour résoudre le problème de stockage de conteneurs. Ils se sont focalisés sur le stockage des conteneurs qui sont en exportation, dans le but de minimiser simultanément le coût total de stockage et la distance totale parcourue par les véhicules internes entre la cour de stockage et les quais. Dans leur algorithme, ils ont considéré deux colonies qui effectuent chacune une tâche prédéfinie : la première affecte les conteneurs de chaque client à un bloc, alors que la deuxième désigne un quai pour chaque groupe de conteneurs qui appartiennent à un même client. Ainsi, ils traitent séquentiellement le problème, en appliquant d'abord la première colonie puis la deuxième sur chaque groupe de conteneurs. Cependant, les auteurs ont utilisé deux matrices différentes de phéromone, dont chacune est spécifique à une colonie. A chaque itération de leur algorithme, ils mettent à jour uniquement les traces de phéromone appartenant aux solutions qui se trouvent sur le front Pareto. Yalaoui et al. n'ont pas proposé de formulations mathématiques dans cet article, mais ils ont utilisé un algorithme d'énumération pour mesurer l'efficacité de leur algorithme de colonie de fourmis. Les auteurs sont allés plus loin en renforçant leur algorithme de colonie de fourmis par un algorithme de recherche locale (l'algorithme résultant est appelé MACO-LS). Pour ce faire, à chaque itération, les solutions non-dominées sont modifiées en changeant les blocs et les quais qui sont affectés à certains groupes de conteneurs tout en respectant les contraintes de capacité. Après avoir effectué des simulations numériques, Yalaoui et al. ont conclu que les deux algorithmes fournissent chacun de très bons résultats, mais MACO-LS est plus performant que MACO.

L'approche proposée dans cet article semble intéressante, cependant les auteurs n'ont pas tenu compte du fait que l'allocation de navires aux postes à quai ne se fait pas en considérant uniquement les conteneurs qui sont en exportation, il se fait surtout en tenant

compte de la disponibilité des postes à quai et aussi des durées de séjour des navires. De ce fait, pour minimiser la distance totale parcourue entre les blocs et les postes à quai, il serait intéressant de traiter le problème à l'envers ; autrement dit : étant donnée une allocation de postes à quai, décider dans quels blocs affecter les conteneurs de sorte à minimiser simultanément les coûts de stockage et la distance totale à parcourir pour transporter ces conteneurs de la cour de stockage vers les quais.

3.4 Conclusion

Dans ce chapitre, nous avons présenté une étude bibliographique sur le problème de stockage de conteneurs dans un terminal portuaire. Différentes stratégies de stockage sont évoquées, ainsi que des méthodes de résolution analytiques et numériques. D'une manière générale, les papiers relatés, dans cette étude bibliographique, considèrent des hypothèses variées qui sont généralement en relation avec les équipements utilisés dans les ports considérés. Certains papiers ont utilisé des techniques d'optimisation, qui sont expliquées dans ce chapitre, pour résoudre le problème de stockage de conteneurs. Ces méthodes d'optimisation peuvent être : exactes, heuristiques, ou méta-heuristiques.

L'étude bibliographique a révélé que le problème de stockage de conteneurs dans un terminal portuaire a été abordé sous différents aspects. Cependant, la majorité des publications proposent des méthodes de stockage par ségrégation, tout en soulignant le fait que cette stratégie est gourmande en surface de stockage. D'autres papiers prônent la stratégie de stockage par groupe, qui alloue un groupe de piles à chaque groupe de conteneurs qui ont des similarités. Le point faible de cette méthode est l'absence de précision dans l'emplacement exact qui est assigné à chaque conteneur.

Au meilleur de nos connaissances, même les rares papiers qui proposent de déterminer avec précision un emplacement de stockage pour chaque conteneur considèrent, soit les conteneurs entrants, soit les conteneurs sortants. Nous allons donc proposé, dans les chapitres suivants, des méthodes de stockage qui déterminent avec précision l'emplacement de stockage affecté à chaque conteneur, et qui prennent en considération les conteneurs entrants ainsi que ceux qui sont sortants ou en transition.

Deuxième partie

Approches proposées

Chapitre 4

Approches proposées pour la résolution du cas statique

4.1 Introduction

Pour résoudre le problème de stockage de conteneurs, on peut distinguer deux cas : statique et dynamique. La principale différence entre ces deux cas est le fait que dans le cas statique, on considère que tous les conteneurs sont déjà arrivés au port avant le début des opérations de stockage ; alors que dans le cas dynamique, on prend en considération les conteneurs qui vont arriver après le début des opérations de stockage.

Dans ce chapitre, nous traitons le cas statique du problème de stockage de conteneurs. Pour ce faire, nous adoptons deux types d'approches : une approche analytique (dans laquelle nous donnons une modélisation mathématique et étudions la complexité du problème) et une approche numérique (en proposant des algorithmes de résolution).

Le reste du chapitre est organisé comme suit. Dans la section 4.2, nous décrivons le contexte, en précisant les hypothèses ainsi que les équipements de manutention et de transfert considérés. Une modélisation mathématique du problème de stockage de conteneurs est donnée dans la section 4.3. La complexité du problème est étudiée dans la section 4.4. Un algorithme de branch-and-cut et un algorithme de colonie d'abeilles sont proposés respectivement dans les sections 4.5 et 4.6. Une conclusion est présentée dans la section 4.7.

4.2 Description du problème

Nous considérons un terminal à conteneurs multi-modal, à l'image de ceux du port du Havre où des conteneurs transitent entre navires, trains et camions. Nous prenons en considération aussi bien les conteneurs entrants (c'est-à-dire qui viennent par la mer et qui partent par voie routière ou ferroviaire), que sortants (qui viennent par voie routière ou ferroviaire et qui partent par la mer) et aussi ceux qui transitent entre différents navires. Les opérations de manutention et de transfert sont réalisées par des cavaliers gerbeurs qui sont capables de transporter chacun un conteneur à la fois. Dans [79], Moussi a considéré un terminal à conteneurs qui utilise des cavaliers gerbeurs, mais il ne s'est

intéressé qu'aux conteneurs entrants en supposant qu'ils ne sont pas mélangés aux autres types (sortant ou en transit) de conteneur dans la cour de stockage. Cela sous-entend que la cour de stockage est divisée en zones qui sont réservées chacune à un type de conteneur précis. L'inconvénient de cette stratégie est que, à un moment donné, l'espace alloué à un type de conteneur peut devenir insuffisant alors que ceux des autres types de conteneur ne sont pas pleins. Pour éviter ce genre de problème, nous considérons que des conteneurs de différents types peuvent être stockés ensemble. Cependant, pour accélérer les opérations de livraison et de chargement des navires et des trains, nous définissons des catégories de conteneur et minimisons la distance qui sépare deux conteneurs de même catégorie. Dans [79], Moussi a considéré un terminal à conteneurs où les remaniements sont complètement interdits, ce qui veut dire que l'on peut placer un conteneur dans une pile seulement si cette dernière est vide ou bien si le conteneur va quitter la cour de stockage avant les conteneurs qui sont déjà à l'intérieur de la pile. Cela n'est réalisable que dans le cas d'un terminal à conteneurs qui dispose, à chaque instant, d'assez de piles vides. Ce qui est de plus en plus rare, car l'augmentation continuelle des capacités des navires porte-conteneurs entraîne en même temps l'augmentation des quantités de conteneurs qui peuvent arriver simultanément dans un port. Ainsi, pour être plus réaliste, nous considérons que les remaniements ne sont pas interdits, mais nous les minimisons. En plus de cela, nous considérons un terminal à conteneurs qui a plusieurs sorties (chacune reliée à un mode de transport), et minimisons les distances prévues entre les emplacements de stockage et les sorties. Ce procédé permet d'accélérer les opérations de chargement des navires et des trains, et de diminuer les temps d'attente des camions.

Dans les méthodes de résolution que nous proposons, nous considérons ces quatre hypothèses suivantes.

1) On tient compte des différences de taille entre les conteneurs, et on stocke dans chaque pile uniquement des conteneurs de mêmes dimensions.

2) Des conteneurs sont de même catégorie s'ils ont les mêmes dimensions et sont destinés à un même train, ou bien à un même navire, ou bien s'ils appartiennent à un même client dans le cas des conteneurs qui partent par voie routière.

3) On suppose que les conteneurs sont numérotés suivant leurs ordres d'arrivée et de déchargement, puisque les plans de déchargement des navires et des trains sont connus à l'avance.

4) On considère que les piles sont numérotées de sorte que deux piles adjacentes qui sont dans une même travée aient des numéros successifs, et que si deux travées sont adjacentes alors le numéro de la dernière pile de l'une succède au numéro de la première pile de l'autre (voir la figure 23).

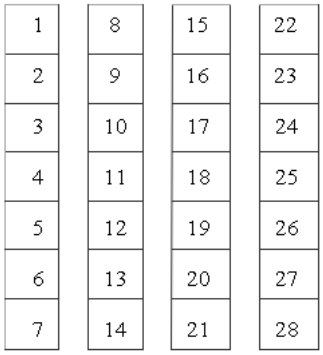


FIGURE 23 – Numérotation des piles

4.3 Modélisation mathématique

Pour la résolution du cas statique du problème de stockage de conteneurs, nous proposons une formulation mathématique du problème sous forme d'un programme linéaire en nombres entiers (PLNE) dans lequel sont utilisés les paramètres décrits dans le tableau 4.1.

Tableau 4.1 – Paramètres

	Notation	Description
Indice	k	Conteneur
	p	Pile
	i	Emplacement dans une pile
Données concernant les piles	N_p	Nombre de piles
	c_p	Nombre d'emplacements libres dans p
	r_p	Dimension de la pile p (20 pieds, ou 40 pieds, ou 45 pieds)
	t_p	Date de départ du conteneur qui est déjà au sommet de p, au début des opérations de stockage
Données concernant les conteneurs	N	Nombre de conteneurs
	R_k	Dimension du conteneur k (20 pieds, ou 40 pieds, ou 45 pieds)
	T_k	Date de départ du conteneur k
	V_k	Catégorie à laquelle appartient le conteneur k
Données générales	d_p^k	Distance entre la pile p et la porte de sortie du conteneur k
	M	un grand nombre entier

La formulation est donnée ci-dessous.

Soient les trois variables de décision suivantes :

$$x_{p,i}^k = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à l'emplacement } i \\ & \text{de la pile } p \\ 0 & \text{Sinon} \end{cases}$$

$$y_p^k = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à la pile } p \\ & \text{et que sa date de départ est supérieure à celle} \\ & \text{d'un conteneur qui est déjà dans } p \\ 0 & \text{Sinon} \end{cases}$$

$z_k^{k'} \in \mathbb{N}$: la distance (en nombre de piles) entre les piles où sont stockés deux conteneurs k et k' appartenant à une même catégorie. Si k et k' n'appartiennent pas à une même catégorie, ou bien s'ils sont stockés dans une même pile, alors $z_k^{k'} = 0$.

Pour le cas statique du problème de stockage de conteneurs, nous proposons le modèle mathématique suivant.

$$\text{Minimiser } f(x,y,z) = \left(\sum_{k=1}^{N} \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} d_p^k x_{p,i}^k, \sum_{k=1}^{N} \sum_{p=1}^{N_p} y_p^k, \sum_{k=1}^{N-1} \sum_{k'=k+1}^{N} z_k^{k'} \right)$$
$$\text{avec } \begin{cases} x = (x_{p,i}^k)_{\substack{1 \le k \le N \\ 1 \le p \le N_p \\ 1 \le i \le c_p}} \\ y = (y_p^k)_{\substack{1 \le p \le N_p \\ 1 \le k \le N}} \\ z = (z_k^{k'})_{\substack{1 \le k \le N \\ k+1 \le k' \le N}} \end{cases} \tag{4.1}$$

La fonction objectif (4.1) minimise simultanément le nombre de remaniements et la distance totale prévue entre les emplacements de stockage et les sorties. De plus, elle minimise les distances de séparation entre les conteneurs qui appartiennent à une même catégorie. Les solutions du problème vérifient les contraintes suivantes.

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} x_{p,i}^k = 1, \ \forall\, k = 1, ..., N \tag{4.2}$$

La contrainte (4.2) assure que chaque conteneur est affecté à un seul emplacement.

$$\sum_{k=1}^{N} x_{p,i}^k \le 1, \ \forall\, p = 1, ..., N_p, \ i = 1, ..., c_p \tag{4.3}$$

La contrainte (4.3) garantit que plusieurs conteneurs ne sont pas simultanément affectés à un même emplacement.

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} x_{p,i}^k = 0, \ \forall\, k = 1, ..., N \ : \ R_k \neq r_p \tag{4.4}$$

La contrainte (4.4) assure que dans chaque pile, sont stockés uniquement des conteneurs qui ont les mêmes dimensions.

$$\sum_{k=1}^{N}(N-k+1)x_{p,i}^{k} \geq \sum_{k=1}^{N}(N-k+1)x_{p,i+1}^{k} \quad \forall\, p=1,...,N_p,\ i=1,...,c_p-1 \tag{4.5}$$

La contrainte (4.5) impose que, dans chaque pile, les conteneurs sont stockés suivant le même ordre que leurs arrivées ou bien leurs déchargements. Dans le cas des conteneurs venant de navires ou de trains, cela permet d'éviter des encombrements sur les quais ou bien au près des rails. Pour le cas des conteneurs apportés par des camions, ce procédé permet de réduire les temps d'attente des camions.

$$y_p^k \leq \sum_{i=1}^{c_p} x_{p,i}^k,\ \forall\, p=1,...,N_p,\ k=1,...,N \tag{4.6}$$

La contrainte (4.6) assure que, si le conteneur k n'est pas affecté à la pile p, alors $y_p^k = 0$. En d'autres termes, si un conteneur n'est pas affecté à une pile, alors il ne peut pas causer de remaniements dans cette pile.

$$(T_k - t_p)\sum_{i=1}^{c_p} x_{p,i}^k \leq M y_p^k \quad \forall\, p=1,...,N_p,\ k=1,...,N \tag{4.7}$$

Si le conteneur k est affecté à la pile p, et que l'un des conteneurs qui sont déjà dans p a une date de départ inférieure à celle de k, alors la contrainte (4.7) force que $y_p^k = 1$, car il y aura forcément un remaniement.

$$\sum_{k=1}^{N} T_k x_{p,i}^k \geq \sum_{k=1}^{N} T_k x_{p,i+1}^k \quad \forall\, p=1,...,N_p,\ i=1,...,c_p-1 \tag{4.8}$$

La contrainte (4.8) assure que dans chaque pile, les nouveaux conteneurs sont stockés suivant l'ordre décroissant de leurs dates de départ. Cela permet de minimiser le nombre de remaniements.

$$z_k^{k'} = \left|\sum_{p=1}^{N_p}\sum_{i=0}^{c_{max}} p x_{p,i}^k - \sum_{p=1}^{N_p}\sum_{i=0}^{c_{max}} p x_{p,i}^{k'}\right| \quad \forall\, k=1,...,N,\ k'=k+1,...,N\ :\ V_k = V_{k'} \tag{4.9}$$

La contrainte (4.9) calcule la distance entre les emplacements de stockage de deux conteneurs qui font partie d'une même catégorie.

$$z_k^{k'} = 0, \quad \forall\, k = 1, ..., N,\ k' = 1, ..., N, \\ \text{tel que } V_k \neq V_{k'} \text{ ou } k \geq k' \tag{4.10}$$

La contrainte (4.10) force que $z_k^{k'} = 0$ si k et k' ne sont pas de même catégorie, ou bien si $k \geq k'$ puisqu'il suffit de traiter le cas où $k < k'$.

$$x_{p,i}^k \in \{0,1\},\ y_p^k \in \{0,1\},\ \text{et } z_k^{k'} \in \mathbb{N} \\ \forall\, p = 1, ..., N_p,\ i = 1, ..., c_p,\ k = 1, ..., N \tag{4.11}$$

La contrainte (4.11) précise la nature de chaque variable de décision.

4.4 Complexité du problème étudié

Pour étudier la complexité du problème de stockage de conteneurs, nous utilisons des notions de théorie des graphes. Plus précisément nous ramenons le problème à un problème de coloration de graphe. Ainsi, par souci de clarté, nous allons, dans un premier temps, rappeler des notions qui sont utiles pour la compréhension de ce qui suit.

4.4.1 Notions préliminaires

Soit $G = (V, E)$ un graphe non orienté, V est l'ensemble des sommets et E est l'ensemble des arêtes.

On dit que G est un graphe de *comparabilité* si et seulement s'il existe un ordre des sommets $v_1, ..., v_n$ de V, tel que pour chaque triplet (p, q, r) vérifiant $p < q < r$, si $(v_p, v_q) \in E$ et $(v_q, v_r) \in E$ alors $(v_p, v_r) \in E$. Un graphe de *co-comparabilité* est le graphe complémentaire d'un graphe de comparabilité.

On dit que G est un graphe de *permutation* si et seulement si il existe un ordre des sommets $1, ..., n$ de V et une permutation σ des sommets tel que pour tout $i, j \in \{1, ..., n\}$ vérifiant $1 \leq i < j \leq n, (i, j) \in E$ si et seulement si $\sigma(i) > \sigma(j)$.

Théorème 1. *[33] Un graphe G est un graphe de permutation si et seulement si G et son complémentaire sont des graphes de comparabilité.*

Un graphe *d'incompatibilité* est un graphe dans lequel chaque arête relie deux sommets incompatibles pour une relation donnée.

Nous finissons ces rappels en présentant le problème de coloration bornée. Étant donnés un graphe non orienté $G = (V, E)$, un ensemble de s couleurs $\{l_1, ..., l_s\}$, un entier C, et un vecteur de poids qui donne le coût d'affectation d'une couleur l_i à un sommet du graphe. Le problème de coloration bornée de poids minimum consiste à déterminer une coloration de poids minimum du graphe G, en utilisant au plus s couleurs et de telle sorte qu'une couleur soit affectée à au plus C sommets du graphe. Concernant le problème de coloration bornée, nous avons les résultats suivants.

Théorème 2. *[13] Le problème de coloration bornée de poids minimum est polynomial dans le cas général, lorsque $C = 2$. Cependant, elle est NP-difficile, lorsque $C \geq 3$.*

Théorème 3. *[55] Le problème de coloration bornée de poids minimum est NP-difficile dans la classe des graphes de permutation, lorsque $C \geq 6$.*

4.4.2 La complexité du problème

Pour étudier la complexité du problème de stockage de conteneurs, nous considérons le cas idéal, dans lequel la cour de stockage est assez grande pour que l'interdiction des remaniements soit possible. En plus de cela, nous considérons la version mono-objective du problème, dans laquelle seule la distance totale à parcourir entre les emplacements de stockage et les sorties est minimisée.

Dans un premier temps, nous considérons que dans une instance quelconque du problème de stockage de conteneurs (PSC), les piles sont initialement toutes vides. Nous discuterons plus tard du cas où toutes les piles de la cour de stockage ne sont pas initialement vides.

Cas où toutes les piles sont initialement vides

Nous introduisons un graphe non orienté $G(K, T, R) = (V, E)$, construit à partir d'une instance quelconque du problème de stockage de conteneurs, où K est l'ensemble des conteneurs alors que T et R sont des vecteurs donnant respectivement la date de départ et la taille de chaque conteneur. Le graphe $G(K, T, R)$ est construit de la manière suivante. Un sommet du graphe correspond à un conteneur. Pour simplifier les notations, on utilisera l'indice k pour désigner à la fois un conteneur et le sommet du graphe qui lui correspond. On a une arête entre deux sommets k et k' si et seulement si les conteneurs correspondants ne peuvent pas être stockés dans une même pile. En d'autres termes, si l'une des conditions suivantes est vérifiée :

- $k < k'$ et $T_k < T_{k'}$, car dans chaque pile les conteneurs doivent être stockés suivant leur ordre d'arrivée au port et l'ordre décroissant de leurs dates de départ,
- $R_k \neq R_{k'}$, car deux conteneurs qui ont des dimensions inégales ne peuvent pas être stockés dans une même pile.

La figure 24 est un exemple de graphe construit à partir d'une instance du PSC.

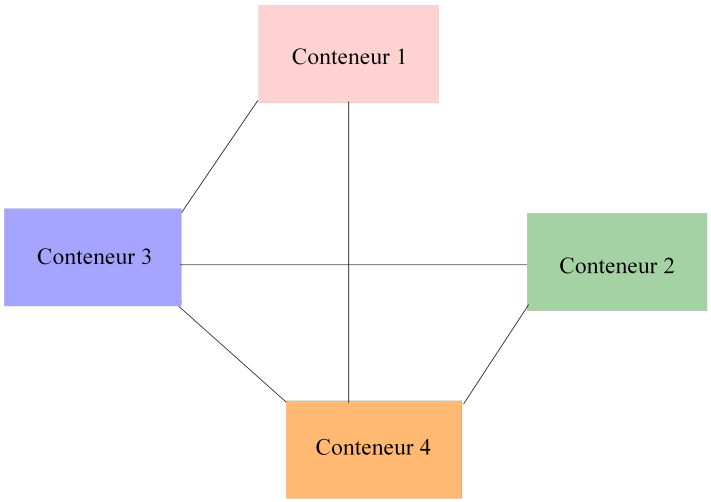


FIGURE 24 – Graphe construit avec une instance du PSC

Ce graphe est construit à partir des données suivantes.

	Dimension	Date de départ
Conteneur 1	1 EVP	10
Conteneur 2	1 EVP	7
Conteneur 3	1 EVP	12
Conteneur 4	2 EVP	17

Dans le cas où les conteneurs ont des tailles égales, on a le résultat suivant.

Lemme 1. *Lorsque tous les conteneurs ont les mêmes dimensions, le graphe $G(K, T, R)$ obtenu à partir d'une instance du PSC est un graphe de permutation.*

Preuve. Tout d'abord, remarquons que lorsque tous les conteneurs sont égaux, il y a une arête entre deux sommets k et k' appartenant à V si et seulement si $k < k'$ et $T_k < T_{k'}$. Pour montrer que $G(K, T, R)$ est un graphe de permutation, il suffit de montrer que lui et son complémentaire sont des graphes de comparabilité (Théorème 1).

Montrons d'abord que $G(K, T, R)$ est un graphe de comparabilité. Considérons trois sommets quelconques k, k' et k'' du graphe vérifiant $k < k' < k''$, tel que $(k, k') \in E$ et que $(k', k'') \in E$. Puisqu'on a $(k, k') \in E$ et $(k', k'') \in E$, alors $k < k' < k''$ et $T_k < T_{k'} < T_{k''}$, ce qui implique que le graphe $G(K, T, R)$ possède une arête entre les sommets k et k''. Donc $G(K, T, R)$ est un graphe de comparabilité.

Nous montrons maintenant que le complémentaire de $G(K, T, R)$, noté $\bar{G}(K, T, R)$, est aussi un graphe de comparabilité. Notons qu'il y a une arête entre deux sommets k

et k' de $\bar{G}(K,T,R)$, si et seulement si k et k' ne sont pas adjacents dans $G(K,T,R)$, c'est-à-dire que $k < k'$ et $T_k \geq T_{k'}$. De même que précédemment, pour trois sommets quelconques du graphe $\bar{G}(K,T,R)$, tel que $k < k' < k''$, $(k,k') \in E$ et $(k',k'') \in E$, on a $T_k \geq T_{k'} \geq T_{k''}$. Donc, $k < k''$ et $T_k \geq T_{k''}$, et il existe une arête entre k et k'' dans $\bar{G}(K,T,R)$. Par conséquent, $\bar{G}(K,T,R)$ est aussi un graphe de comparabilité. □

Maintenant, on montre qu'une solution du problème de stockage de conteneurs correspond à une solution du problème de coloration bornée, et vice versa. En fait, un résultat similaire est donné dans [13]. Considérons une instance $IPSC = (K,T,R,P,C_{max},d)$ du PSC, et le graphe $G(K,T,R)$ qui lui est associé. P est l'ensemble des piles, C_{max} est le nombre maximal de conteneurs qui peuvent être stockés dans une pile, et d est la matrice des distances (matrice des coûts). Considérons maintenant une C_{max}-coloration de $G(K,T,R)$ comprenant s couleurs. On fait correspondre une couleur du problème de coloration bornée à une pile dans le PSC. Rappelons qu'on est dans le cas où les conteneurs sont tous de même mesure et que l'on considère uniquement les piles qui ont les mesures adéquates. Ainsi, comme tous les sommets qui ont une même couleur forment un stable, c'est-à-dire qu'ils ne sont pas reliés par des arêtes, deux conteneurs quelconques correspondants à deux sommets de ce stable sont tels que $k < k'$ et $T_k \geq T_{k'}$. L'ordre d'arrivée et les dates de départ des conteneurs correspondants aux sommets de chaque stable sont compatibles et donc les conteneurs de chaque stable peuvent être stockés dans une même pile. De plus, il y a au maximum C_{max} sommets dans chaque stable. Donc le nombre de conteneurs affectés à chaque pile est bien inférieur ou égal à C_{max}. Une C_{max}-coloration correspond donc bien à une affectation valide pour le problème de stockage de conteneurs. De la même manière, une solution du PSC correspond à une solution du problème de coloration C_{max}-bornée dans le graphe $G(K,T,R)$. Nous en déduisons le Lemme suivant.

Lemme 2. *Soit $IPSC = (K,T,R,P,C_{max},d)$ une instance du problème de stockage de conteneurs. Le PSC pour l'instance IPSC admet une solution si et seulement si le problème de coloration C_{max}-bornée sur le graphe $G(K,T,R)$ admet une solution.*

Nous donnons maintenant le résultat principal de cette section.

Théorème 4. *Dans le cas où les conteneurs sont de même taille et que toutes les piles sont initialement vides, le problème de stockage de conteneurs est équivalent au problème de coloration bornée de poids minimum (PCB).*

Preuve. Pour établir ce résultat, nous montrons qu'une instance du PSC correspond à une instance du PCB et vice versa.

Soient $IPSC = (K,T,R,P,C_{max},d)$ une instance du problème de stockage de conteneurs, et $G(K,T,R)$ le graphe de permutation qui lui est associé. Considérons une instance du problème de coloration bornée $IPCB = (G(K,T,R),C_{max},P,d)$, qui est définie sur le graphe $G(K,T,R)$ avec P couleurs, une borne C_{max}, et une matrice de poids d. D'après le Lemme 2, une solution du PSC correspond à une solution du PCB de même qu'une pile du PSC correspond à une couleur du PCB, et vice versa. Il s'en suit alors que le coût d'affectation d_p^k d'un conteneur $k \in K$ à une pile $p \in P$, est le même que celui d'associer au sommet k la couleur correspondante à la pile p. Donc, le coût d'une C_{max}-coloration

dans le graphe $G(K, T, R)$ est le même que celui de la solution du PSC correspondant, et vice versa. Par conséquent, on peut trouver la solution optimale du PSC si et seulement si on trouve la solution optimale du BCP. □

D'après le Théorème 3, le problème de coloration bornée est NP-difficile lorsque le graphe G est un graphe de permutation et que $C_{max} \geq 6$. On déduit donc, à partir des Théorèmes 4, 2, et 3, que le problème de stockage de conteneurs est NP-difficile.

Dans le cas où les conteneurs ont des dimensions variées, le PSC peut être résolu en considérant plusieurs sous-problèmes, chacun concernant une taille de conteneur précise. Ainsi, une solution du problème peut être obtenue en déterminant de manière indépendante les affectations des conteneurs de chaque taille aux piles qui sont compatibles. Par exemple, si on note K_{1EVP}, K_{2EVP}, P_{1EVP}, et P_{2EVP} les ensembles de conteneurs et de piles de tailles {1 EVP (20 pieds), 2 EVP (40 pieds)}, respectivement ; alors, trouver la solution optimale du problème global revient à déterminer la solution optimale de chaque sous problème (PSC_{1EVP} et PSC_{2EVP}) séparément pour les ensembles de conteneurs et de piles (K_{1EVP}, P_{1EVP}) et (K_{2EVP}, P_{2EVP}), comme le montre la figure suivante.

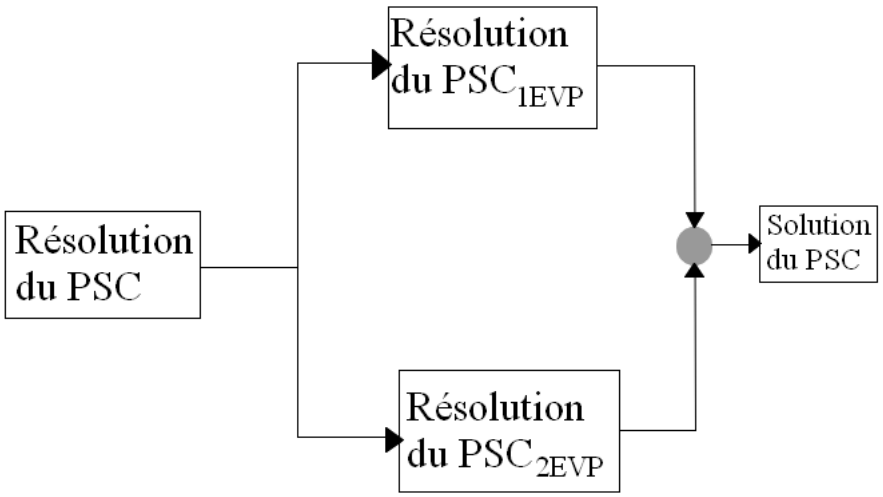


FIGURE 25 – Décomposition du problème

On observe que dans chaque sous-problème les conteneurs et les pilles ont les mêmes mesures, et que les graphes G_i, $i \in \{1EVP, 2EVP\}$, sont des graphes de permutation (Lemme 1). Donc chaque sous-problème est NP-difficile, lorsque $C_{max} \geq 6$.

Corollaire 1. *Le problème de stockage de conteneurs est NP-difficile, lorsque $C_{max} \geq 6$, même si les conteneurs ont des tailles variées.*

Cas où toutes les piles ne sont pas initialement vides

Dans le cas où toutes les piles ne sont pas initialement vides, le PSC est équivalent à un problème de coloration avec capacité [13], qui généralise le problème de coloration bornée (donc qui est aussi NP-difficile). Dans le problème de coloration avec capacité, on

fixe une capacité c_p pour chaque couleur p, et on impose que chaque couleur soit utilisée au plus c_p fois dans une solution. Les capacités c_p peuvent avoir des valeurs différentes.

Pour une instance du PSC où chaque pile $p \in P$ a une capacité c_p, avec $c_p \leq C_{max}$, le PSC devient un problème de coloration avec capacité dans le graphe $G(K, T, R)$ où chaque couleur peut être utilisée au plus c_p fois.

Cas sans incompatibilité

Si on considère un cas particulier du PSC où les ordres d'arrivée et les dates de départ des conteneurs sont tous compatibles, alors les graphes $G_i,\ i \in \{1EVP, 2EVP\}$ ne contiennent aucune arête. Trouver une solution du problème revient donc à trouver une coloration C_{max}-bornée dans un graphe où tous les sommets sont isolés. Il est donc équivalent à un problème d'affectation où les sommets du graphe sont les objets à affecter et les couleurs sont les ressources auxquelles les objets sont affectés, et où chaque ressource se voit affectée à au plus C_{max} objets. Le PSC est donc polynomial dans ce cas.

Corollaire 2. *Le problème de stockage de conteneurs peut être résolu en temps polynomial lorsque les dates de départ et les ordres d'arrivée et de déchargement des conteneurs sont tous compatibles.*

4.5 Algorithme de branch-and-cut pour la résolution du problème de stockage de conteneurs

Dans le cas où la cour de stockage est assez grande pour que le stockage sans aucun remaniement soit possible, nous proposons un algorithme de branch-and-cut pour la résolution du problème de stockage de conteneurs. Pour ce faire, nous utilisons le graphe d'incompatibilité décrit dans la section 4.4.2, où les sommets représentent les conteneurs, et où chaque arête relie deux conteneurs qui ont des tailles différentes ou bien dont les ordres d'arrivée et de départ ne sont pas compatibles. Puisque nous savons que deux conteneurs adjacents dans le graphe ne peuvent pas être stockés ensemble dans une pile, nous exploitons cette propriété et utilisons dans notre algorithme de branch-and-cut le modèle mathématique simplifié suivant.

Les paramètres utilisés dans ce modèle sont décrits dans le Tableau 4.1, mais la variable de décision utilisée est la suivante.

$$x_p^k = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à la pile } p \\ 0 & \text{Sinon} \end{cases}$$

Comme on peut le remarquer, cette variable de décision précise la pile à laquelle est affecté chaque conteneur, mais elle ne précise pas l'emplacement exact d'un conteneur dans une pile. Cela ne pose pas de problème, car la manière dont est construit le graphe assure que les ordres d'arrivée et de départ des conteneurs qui sont affectés à une même pile sont compatibles. L'absence de remaniements et la minimisation des distances à parcourir entre les emplacements de stockage et les portes de sortie des conteneurs entraînent une

accélération des activités de chargement (de navires, trains, et camions). Par conséquent, le regroupement des conteneurs par catégorie n'est plus indispensable, et donc nous considérons la version mono-objective du problème de stockage de conteneurs décrite comme suit.

$$Minimiser \sum_{k=1}^{N} \sum_{p=1}^{N_p} d_p^k x_p^k \tag{4.1}$$

La fonction objectif (4.1) minimise la distance totale à parcourir entre les emplacements de stockage et les portes de sortie des conteneurs.

$$\sum_{p=1}^{N_p} x_p^k = 1, \quad \forall\, k = 1, ..., N \tag{4.2}$$

La contrainte (4.2) assure que chaque conteneur est affecté à une seule pile.

$$x_p^k + x_p^{k'} \leq 1, \quad \forall\, (k, k') \in E,\ p = 1, ..., N_p \tag{4.3}$$

La contrainte (4.3) garantit que des conteneurs incompatibles ne sont pas affectés à une même pile. E est l'ensemble des arêtes du graphe d'incompatibilité.

$$\sum_{k=1}^{N} x_p^k \leq c_p, \quad \forall\, p = 1, ..., N_p \tag{4.4}$$

La contrainte (4.4) force que le nombre de conteneurs affectés à chaque pile est inférieur où égal au nombre d'emplacements libres de la pile.

$$\sum_{k=1}^{N} x_p^k = 0, \quad \forall\, p = 1, ..., N_p : T_k > t_p \text{ ou } R_k \neq r_p \tag{4.5}$$

La contrainte (4.5) assure qu'un conteneur n'est affecté à une pile que si sa taille est compatible avec celle de la pile, et que sa date de départ est inférieure à celles des conteneurs qui sont déjà stockés dans la pile au début de la période de stockage courante.

$$x_p^k \in \{0, 1\}, \quad \forall\, k = 1, ..., N,\ p = 1, ..., N_p \tag{4.6}$$

La contrainte (4.6) précise que les variables de décision sont booléennes.

Dans la section suivante, nous allons expliquer le principe de l'algorithme de branch-and-cut que nous proposons pour la résolution du problème de stockage de conteneurs.

4.5.1 Description de l'algorithme

Le branch-and-cut est une méthode de résolution exacte qui combine la méthode de branch-and-bound et celle du coupe de plan. Chacune de ces méthodes fonctionne en résolvant une séquence de relaxations du problème en nombres entiers initial. Les méthodes de coupe de plan améliorent la relaxation du problème en nombres entiers, afin de trouver une meilleure approximation, alors que les algorithmes de branch-and-bound

utilisent une approche sophistiquée appelée *diviser pour régner*.

L'algorithme de branch-and-cut utilise un arbre de recherche dont la racine est une relaxation du problème mixte en nombres entiers à résoudre. Les autres nœuds de l'arbre de recherche sont créés en faisant des partitions de l'espace de recherche de solution, autrement dit en faisant des branchements. La majeure différence entre le branch-and-cut et le branch-and-bound est le fait que la première méthode utilise des inégalités valides (plans de coupe) pour améliorer la solution trouvée à chaque nœud de l'arbre de recherche, avant de créer de nouvelles branches. Une inégalité valide est une inégalité qui doit être vérifiée par toute solution du problème mixte en nombres entiers, mais qui n'est pas forcément satisfaite par les solutions des sous-problèmes relaxés.

Pour effectuer l'algorithme de branch-and-cut, il est crucial d'avoir : une ou plusieurs inégalités valides, une méthode de relaxation du problème étudié, une technique de recherche d'une borne supérieure, une règle de branchement, et une méthode de recherche d'inégalités valides qui ne sont pas satisfaites (appelée aussi méthode de séparation).

Nous allons, dans un premier temps, détailler ces dernières, avant de décrire la succession d'étapes de notre algorithme de branch-and-cut.

Création d'une inégalité valide

Soit k un sommet (conteneur) du graphe, tel que $1 \leq k \leq N$. Soit $N(k)$ l'ensemble des voisins de k, et soit p une couleur (pile), tel que $1 \leq p \leq N_p$.

En additionnant les contraintes (4.3) sur le voisinage $N(k)$, on obtient l'inégalité valide suivante

$$\sum_{k' \in N(k)} x_p^{k'} + |N(k)| x_p^k \leq |N(k)|, \quad \forall\, k = 1, ..., N;\ \text{et}\ p = 1, ..., N_p \tag{4.7}$$

Proposition 1. *Dans la description des solutions du problème de coloration bornée, les inégalités (4.3) et (4.7) sont équivalentes.*

Preuve. Il suffit de prouver que (4.7) implique (4.3), car le sens inverse est mis en évidence par la définition de (4.7).

Soit $x \in \{0,1\}^{N \times N_p}$ une solution du problème de coloration bornée.

- Si $x_p^k = 1$, alors $\sum_{k' \in N(k)} x_p^{k'} = 0$. Par conséquent, pour tout k' voisin de k, on a $x_p^{k'} = 0$. Donc $x_p^k + x_p^{k'} = 1,\ \forall\, k' \in N(k)$.
- Si $x_p^k = 0$, alors $\sum_{k' \in N(k)} x_p^{k'} \leq |N(k)|$. Ce qui signifie que, pour tout k' appartenant à $N(k)$, $x_p^{k'}$ peut être égal, soit à 0, soit à 1. Ainsi, $x_p^{k'} + x_p^k \leq 1,\ \forall\, k' \in N(k)$. □

Relaxation du problème

Généralement, pour effectuer un algorithme de Branch-and-Cut, on procède à la *relaxation linéaire* de la formulation du problème, en éliminant les contraintes d'intégrité des variables de décision. Cependant, dans notre cas, en plus de la relaxation linéaire du problème (la contrainte (4.6) devient $x_p^k \geq 0,\ \forall\, k = 1, ..., N;\ p = 1, ..., N_p$), nous enlevons

les contraintes d'adjacence (4.3), pour diminuer le nombre de contraintes du modèle mathématique. Cela n'affecte pas l'optimalité des solutions fournies par notre algorithme de branch-and-cut, car des inégalités valides de la forme (4.7) sont utilisées, à chaque noeud de l'arbre de recherche, pour assurer l'exactitude des solutions.

Méthode de recherche d'une borne supérieure (Heuristique primale)

La solution du modèle relaxé nous donne une borne inférieure de la solution du programme linéaire entier. Cependant, pour accélérer l'algorithme de branch-and-cut, il est utile d'avoir aussi une borne supérieure. Pour obtenir cette dernière, nous utilisons un algorithme heuristique qui colorie, un par un, les sommets du graphe, suivant l'ordre décroissant de leurs nombres de voisins non coloriés. Pour chaque sommet, on choisit parmi les couleurs admissibles (c'est-à-dire qui ne sont pas attribuées à des sommets voisins du sommet à colorier, qui ont la bonne dimension, et qui satisfont la contrainte de date) celle qui correspond à la pile la plus proche de la porte de sortie du conteneur correspondant au sommet en question. Des mises à jour sont effectuées sur les capacités des piles au fur et à mesure que l'algorithme progresse.

Règle de branchement

Lorsque la solution fournie par un nœud de l'arbre de recherche n'est pas entière, on peut l'améliorer en effectuant un branchement. Pour ce faire, nous choisissons la variable de décision la plus fractionnaire, c'est-à-dire celle qui est plus à mi-chemin entre sa partie entière inférieure et sa partie entière supérieure. Par exemple, si on a $x = (1.25, 3.45, 2.75, 5)$, alors on branche sur 3.45, car 0.45 est plus proche de $\frac{1}{2}$ que 0.25, ou 0.75, ou bien 0.

L'algorithme de recherche de la variable de décision la plus fractionnaire est comme suit.

1. $select = 1$;
2. Pour $(k = 1, ..., N)$ faire
   - 2.a. Pour (p=1,...,N_p)
     - 2.a.1. Si $(|\lfloor x_p^k \rfloor + \frac{1}{2} - x_p^k| < select)$
       - $select = x_p^k$
     - 2.a.2 Fin Si
   - 2.b. Fin Pour
3. Fin Pour

Pour effectuer un branchement, nous utilisons la règle classique de branchement, en créant deux sous-problèmes. Soit x_p^k la variable la plus fractionnaire. On pose $x_p^k = 0$ dans une branche, cela signifie que le sommet k n'aura pas la couleur p dans cette branche (autrement dit, le conteneur k ne sera pas affecté à la pile p dans cette branche). Puis, dans l'autre branche, on pose $x_p^k = 1$, ce qui veut dire que le sommet k aura forcément la couleur p dans cette branche (en d'autres termes, le conteneur k sera forcément affecté à la pile p dans cette branche).

Séparation des contraintes (4.7)

Une méthode de séparation sert à détecter des inégalités valides qui ne sont pas vérifiées par une solution d'un sous-problème relaxé (nœud de l'arbre de recherche) du problème entier. Ainsi, à chaque nœud de l'arbre de recherche, avant d'effectuer un branchement, nous utilisons un algorithme simple pour chercher des inégalités de voisinage qui sont violées. Pour ce faire, nous traitons une par une toutes les variables qui ont des valeurs supérieures à 0.5 dans la solution optimale du nœud courant. Soient x_p^k une de ces variables, et S un entier initialement nul. Nous calculons le nombre n de voisins du sommet k. Ensuite, nous ajoutons à S la valeur de x_p^k multipliée par n. Puis, nous cherchons toutes les variables $x_{p'}^{k'}$ qui sont tel que k et k' soient voisins et $p' = p$, nous ajoutons la somme de leurs valeurs à S. Si S devient supérieur à n, alors on a une inégalité violée donc on ajoute au sous-problème une contrainte qui représente cette inégalité pour garantir la satisfaction des contraintes (4.7).

Tableau 4.3: Algorithme de branch-and-cut

1. *Initialisation :*	Soit P^0 le problème initial en nombre entier. Initialiser la liste des nœuds actifs de l'arbre de recherche, $T = \{P^0\}$. Utiliser l'algorithme décrit précédemment pour chercher une borne supérieure **BS**. S'il ne trouve pas de solution, alors poser **BS**$=+\infty$.
2. *Condition d'arrêt :*	Si $T = \emptyset$, alors la solution optimale est celle dont la valeur est égale à **BS**. Si **BS**$=+\infty$, alors il n'y a pas de solution.
3. *Choix d'un nœud :*	Choisir dans T un nœud P^l à explorer, qui sera ensuite supprimé.
4. *Relaxation :*	Résoudre avec CPLEX la relaxation de P^l. S'il n'y a pas de solution, alors poser $\mathbf{BI}_l = +\infty$ et aller à l'étape 6. Sinon, noter S^{Rl} la solution optimale de P^l, et utiliser sa valeur pour mettre à jour $\mathbf{BI}_l$.
5. *Ajouter des inégalités valides :*	Rechercher des inégalités valides qui sont violées par S^{Rl}. S'il y en a, alors ajouter les à la relaxation de P^l et retourner à l'étape 4.
6. *Pénétration et élagage :*	(a) Si $\mathbf{BI}_l \geq$**BS** alors retourner à l'étape 2. (b) Si $\mathbf{BI}_l <$**BS** et S^{Rl} est une solution entière réalisable, alors mettre à jour **BS**$=\mathbf{BI}_l$, ensuite supprimer de T tout élément j dont la borne inférieure vérifie $\mathbf{BI}_j \geq$**BS**, puis retourner à l'étape 2.
7. *Branchement :*	Si S^{Rl} est fractionnaire, alors effectuer un branchement sur la variable la plus fractionnaire. On obtient ainsi deux sous-problèmes, $P^{l,1}$ et $P^{l,2}$, que l'on ajoute dans T

À l'étape 3 de l'algorithme, il est possible d'utiliser l'une de ces méthodes suivantes pour choisir un nœud à explorer : méthode "opportuniste", parcours en largeur d'abord, ou bien parcours en profondeur d'abord.

La méthode “opportuniste” choisit parmi les nœuds qui ne sont pas encore explorés, celui qui a la plus petite borne inférieure (notons que la borne inférieure est la valeur de fonction objectif, obtenue après le traitement du nœud père).

La méthode du parcours en largeur d'abord essaie toujours d'explorer le nœud le plus proche de la racine de l'arbre de recherche. Ainsi, la racine est le premier nœud visité, suivi des nœuds qui sont issus de la racine, etc. La figure 26 en est un exemple.

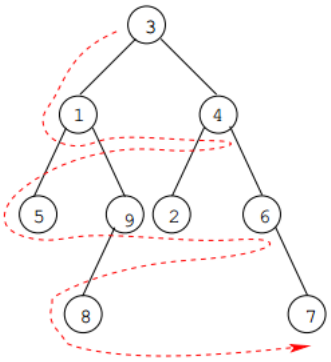


FIGURE 26 – Exemple de parcours en largeur d'abord

Les nœuds sont parcourus dans cet ordre : 3, 1, 4, 5, 9, 2, 6, 8, 7.

Le parcours en profondeur d'abord privilégie les nœuds qui sont plus éloignés de la racine, qui sont issus de pères déjà explorés, comme le montre l'exemple donné dans la figure 27

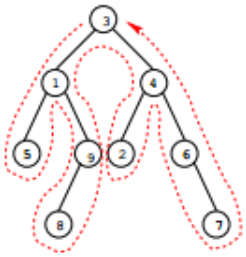


FIGURE 27 – Exemple de parcours en profondeur d'abord

Les nœuds sont parcourus dans cet ordre : 3, 1, 5, 9, 8, 4, 2, 6, 7.

4.5.2 Application de l'algorithme de branch-and-cut sur un exemple

Considérons le stockage de 5 conteneurs de même taille dans 3 pilles qui sont initialement vides. Dans cet exemple, on suppose que les conteneurs sont égaux et que la capacité maximale d'une pile est égale à 3. Les dates de départ des conteneurs sont mentionnées dans le tableau 4.4.

Tableau 4.4 – Données

Conteneur	Date de départ
1	10
2	13
3	8
4	16
5	10

Les distances utilisées dans l'exemple sont notées dans le tableau 4.6.

Tableau 4.6 – Matrice des distances

p	d_p^1	d_p^2	d_p^3	d_p^4	d_p^5
1	105	378	205	378	400
2	118	391	118	291	413
3	106	378	165	332	314

Le graphe d'incompatibilité obtenu à partir des données du tableau 4.4 est représenté dans la figure 28.

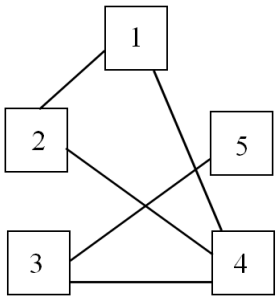


FIGURE 28 – Graphe d'incompatibilité

Dans un premier temps, nous utilisons l'algorithme heuristique décrit dans la section 4.5.1 pour rechercher une borne supérieure (**BS**) du problème de stockage de conteneurs. Pour ce faire, nous commençons avec le conteneur 4, parce qu'il a plus de voisins dans le graphe, nous l'affectons à la pile 2 qui est plus proche de sa porte de sortie. Après cela, nous remarquons que chacun des conteneurs restants a un seul voisin non colorié. Par conséquent, nous réalisons les affectations suivantes dans le même ordre : conteneur 1 à la pile 1, conteneur 3 à la pile 3, conteneur 2 à la pile 3, et le conteneur 5 à la pile 1. La borne supérieure correspondante est : **BS**=1339 ($=d_2^4 + d_1^1 + d_3^3 + d_3^2 + d_1^5$).

À partir du graphe de la figure 28, des données du tableau 4.6, et celles du tableau 4.4, nous construisons le programme linéaire suivant.

$$P^0 = \begin{cases} \text{Minimiser} \\ d : 105x_1^1 + 118x_2^1 + 106x_3^1 + 378x_1^2 + 391x_2^2 + \\ \quad 378x_3^2 + 205x_1^3 + 118x_2^3 + 165x_3^3 + 378x_1^4 + \\ \quad 291x_2^4 + 332x_3^4 + 400x_1^5 + 413x_2^5 + 314x_3^5 \\ \text{Sous contraintes} \\ (1) : x_1^1 + x_2^1 + x_3^1 = 1 \\ (2) : x_1^2 + x_2^2 + x_3^2 = 1 \\ (3) : x_1^3 + x_2^3 + x_3^3 = 1 \\ (4) : x_1^4 + x_2^4 + x_3^4 = 1 \\ (5) : x_1^5 + x_2^5 + x_3^5 = 1 \\ (6) : x_1^1 + x_1^4 \leq 1 \\ (7) : x_2^1 + x_2^4 \leq 1 \\ (8) : x_3^1 + x_3^4 \leq 1 \\ (9) : x_1^1 + x_1^2 \leq 1 \\ (10) : x_2^1 + x_2^2 \leq 1 \\ (11) : x_3^1 + x_3^2 \leq 1 \\ (12) : x_1^2 + x_1^4 \leq 1 \\ (13) : x_2^2 + x_2^4 \leq 1 \\ (14) : x_3^2 + x_3^4 \leq 1 \\ (15) : x_1^3 + x_1^4 \leq 1 \\ (16) : x_2^3 + x_2^4 \leq 1 \\ (17) : x_3^3 + x_3^4 \leq 1 \\ (18) : x_1^3 + x_1^5 \leq 1 \\ (19) : x_2^3 + x_2^5 \leq 1 \\ (20) : x_3^3 + x_3^5 \leq 1 \\ (21) : x_1^1 + x_1^2 + x_1^3 + x_1^4 + x_1^5 \leq 3 \\ (22) : x_2^1 + x_2^2 + x_2^3 + x_2^4 + x_2^5 \leq 3 \\ (23) : x_3^1 + x_3^2 + x_3^3 + x_3^4 + x_3^5 \leq 3 \\ (24) : x_p^k \in \{0, 1\}, \ \forall \ k = 1, ..., 5; \ p = 1, 2, 3 \end{cases}$$

Nous initialisons l'arbre de recherche $T = \{P^0\}$. Ensuite nous relaxons P^0, puis nous obtenons R^0 qui est comme suit.

$$R^0 = \begin{cases} \text{Minimiser} \\ d : 105x_1^1 + 118x_2^1 + 106x_3^1 + 378x_1^2 + 391x_2^2 + \\ \quad 378x_3^2 + 205x_1^3 + 118x_2^3 + 165x_3^3 + 378x_1^4 + \\ \quad 291x_2^4 + 332x_3^4 + 400x_1^5 + 413x_2^5 + 314x_3^5 \\ \text{Sous contraintes} \\ (1) : x_1^1 + x_2^1 + x_3^1 = 1 \\ (2) : x_1^2 + x_2^2 + x_3^2 = 1 \\ (3) : x_1^3 + x_2^3 + x_3^3 = 1 \\ (4) : x_1^4 + x_2^4 + x_3^4 = 1 \\ (5) : x_1^5 + x_2^5 + x_3^5 = 1 \\ (21) : x_1^1 + x_1^2 + x_1^3 + x_1^4 + x_1^5 \leq 3 \\ (22) : x_2^1 + x_2^2 + x_2^3 + x_2^4 + x_2^5 \leq 3 \\ (23) : x_3^1 + x_3^2 + x_3^3 + x_3^4 + x_3^5 \leq 3 \\ (24) : x_p^k \geq 0, \ \forall \ k = 1, ..., 5; \ p = 1, 2, 3 \end{cases}$$

Après cela, nous résolvons R^0 avec CPLEX (version 12.6) et obtenons ce résultat ($x_1^1 = 1, x_2^1 = 0, x_1^2 = 1, x_2^2 = 0, x_1^3 = 0, x_2^3 = 1, x_1^4 = 0, x_2^4 = 1, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0, x_3^2 = 0, x_3^3 = 0, x_3^4 = 0, x_3^5 = 1$), qui vaut 1206. Cette solution n'est pas réalisable, car elle ne satisfait pas les inégalités valides suivantes.

$(25) : x_1^2 + x_1^4 + 2x_1^1 <= 2$
$(26) : x_1^1 + x_1^4 + 2x_1^2 <= 2$
$(27) : x_2^4 + x_2^5 + 2x_2^3 <= 2$
$(28) : x_2^1 + x_2^2 + x_2^3 + 3x_2^4 <= 3$

Nous ajoutons alors les inégalités (25), (26), (27), et (28) dans R^0. Ensuite nous le résolvons encore avec CPLEX et obtenons ($x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 0, x_1^3 = 0, x_2^3 = 0.6, x_1^4 = 0, x_2^4 = 0.8, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0, x_3^2 = 1, x_3^3 = 0.4, x_3^4 = 0.2, x_3^5 = 0$), qui vaut 1233. C'est une solution fractionnaire qui ne satisfait pas les inégalités valides suivantes.

$(29) : x_3^1 + x_3^4 + 2x_3^2 <= 2$
$(30) : x_3^3 + x_3^5 <= 1$

Nous ajoutons les inégalités (29) et (30) dans R^0, puis nous le résolvons à nouveau avec CPLEX. Nous obtenons alors ($x_1^1 = 0.666666666666667, x_2^1 = 0, x_1^2 = 0.666666666666667, x_2^2 = 0, x_1^3 = 0, x_2^3 = 1, x_1^4 = 0, x_2^4 = 0, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0.333333333333333, x_3^2 = 0.333333333333333, x_3^3 = 0, x_3^4 = 1, x_3^5 = 1$), qui vaut 1247.3333333333. Cette solution viole l'inégalité valide suivante.

$(31) : x_3^1 + x_3^2 + x_3^3 + 3x_3^4 <= 3$

Après avoir ajouté (31) dans R^0, nous le résolvons encore. Ce qui nous donne ($x_1^1 = 0.75, x_2^1 = 0, x_1^2 = 0.5, x_2^2 = 0, x_1^3 = 0.125, x_2^3 = 0.875, x_1^4 = 0, x_2^4 = 0.25, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0.25, x_3^2 = 0.5, x_3^3 = 0, x_3^4 = 0.75, x_3^5 = 1$), qui vaut 1247.875. Cette solution ne viole aucune inégalité valide, mais elle n'est pas entière. Par conséquent, nous faisons un branchement sur la variable x_1^2 , et donc nous obtenons deux sous-problèmes $P^{00} = P^0 \cup \{x_1^2 = 0\}$ et $P^{01} = P^0 \cup \{x_1^2 = 1\}$, que nous ajoutons dans T. Après cela, nous supprimons P^0, et la liste des nœuds actifs devient $T = \{P^{00}, P^{01}\}$.

Puisque T n'est pas encore vide, nous continuons l'algorithme en sélectionnant P^{00} dans T, nous le relaxons et obtenons R^{00}. Par la suite, nous résolvons R^{00} avec CPLEX, ce qui nous donne ($x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 0.125, x_1^3 = 0.375, x_2^3 = 0.625, x_1^4 = 0, x_2^4 = 0.75, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0, x_3^2 = 0.875, x_3^3 = 0, x_3^4 = 0.25, x_3^5 = 1$), qui vaut 1250.5. Cette solution ne viole aucune inégalité valide, donc nous faisons un branchement sur x_1^3. Ce qui nous donne deux nouveaux sous-problèmes $P^{000} = P^{00} \cup \{x_1^3 = 0\}$ et $P^{001} = P^{00} \cup \{x_1^3 = 1\}$. Nous ajoutons ces deux sous-problèmes dans T, ensuite nous supprimons P^{00}, et donc $T = \{P^{01}, P^{000}, P^{001}\}$.

Nous choisissons P^{01}, le relaxons et le résolvons, ce qui nous donne ($x_1^1 = 0, x_2^1 = 0, x_1^2 = 1, x_2^2 = 0, x_1^3 = 0.166666666666667, x_2^3 = 0.833333333333333, x_1^4 = 0, x_2^4 = 0.333333333333333, x_1^5 = 0, x_2^5 = 0, x_3^1 = 1, x_3^2 = 0, x_3^3 = 0, x_3^4 = 0.666666666666667, x_3^5 =$

1), qui vaut 1248.83333333333. Cette solution ne satisfait pas l'inégalité valide suivante.

(32) : $x_3^2 + x_3^4 + 2x_3^1 <= 1$

Nous ajoutons l'inégalité (32) dans R^{01}, que nous résolvons à nouveau. Ensuite nous obtenons $(x_1^1 = 0, x_2^1 = 0.125, x_1^2 = 1, x_2^2 = 0, x_1^3 = 0.375, x_2^3 = 0.625, x_1^4 = 0, x_2^4 = 0.75, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0.875, x_3^2 = 0, x_3^3 = 0, x_3^4 = 0.25, x_3^5 = 1)$, qui vaut 1251.375. Cette solution ne viole aucune inégalité valide, donc nous branchons sur x_1^3. Ensuite, nous obtenons deux sous-problèmes, $P^{010} = P^{01} \cup \{x_1^3 = 0\}$ et $P^{011} = P^{01} \cup \{x_1^3 = 1\}$. Puis, nous supprimons P^{01}. Ainsi nous obtenons $T = \{P^{000}, P^{001}, P^{010}, P^{011}\}$.

Nous choisissons P^{000}, nous le relaxons puis le résolvons avec CPLEX. Après cela, nous obtenons $(x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 0.375, x_1^3 = 0, x_2^3 = 0.875, x_1^4 = 0, x_2^4 = 0.25, x_1^5 = 0.125, x_2^5 = 0, x_3^1 = 0, x_3^2 = 0.625, x_3^3 = 0.125, x_3^4 = 0.75, x_3^5 = 0.875)$, qui vaut 1258.25. Cette solution ne viole aucune inégalité valide, donc nous branchons sur $x_2^2 = 0.375$. Ainsi, nous obtenons deux nouveaux sous-problèmes : $P^{0000} = P^{000} \cup \{x_2^2 = 0\}$ et $P^{0001} = P^{000} \cup \{x_2^2 = 1\}$. Nous supprimons P^{000}, après cela, nous ajoutons dans T les deux nouveaux sous problèmes, ensuite nous obtenons $T = \{P^{001}, P^{010}, P^{011}, P^{0000}, P^{0001}\}$.

Nous choisissons P^{001}, nous le relaxons puis le résolvons avec CPLEX. Ce qui nous donne $(x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 0, x_1^3 = 1, x_2^3 = 0, x_1^4 = 0, x_2^4 = 1, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0, x_3^2 = 1, x_3^3 = 0, x_3^4 = 0, x_3^5 = 1)$, qui vaut 1293. Cette solution est entière et ne viole aucune inégalité valide. Par conséquent, nous mettons à jour la borne supérieure : **UB**=1293. Après cela, nous supprimons P^{001}, et on obtient $T = \{P^{010}, P^{011}, P^{0000}, P^{0001}\}$.

Nous choisissons P^{010}, nous le relaxons puis le résolvons avec CPLEX. Après cela, nous obtenons $(x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 0, x_1^3 = 1, x_2^3 = 0, x_1^4 = 0, x_2^4 = 1, x_1^5 = 0, x_2^5 = 0, x_3^1 = 0, x_3^2 = 1, x_3^3 = 0, x_3^4 = 0, x_3^5 = 1)$, qui vaut 1293. C'est une solution entière, qui ne viole aucune inégalité valide, mais qui n'améliore pas la borne supérieure courante. Par conséquent, nous supprimons P^{010}, ce qui nous donne $T = \{P^{011}, P^{0000}, P^{0001}\}$.

Nous choisissons P^{011}. Ensuite, nous le relaxons. Puis, nous le résolvons avec CPLEX. Après cela, nous obtenons $(x_1^1 = 0, x_2^1 = 0, x_1^2 = 1, x_2^2 = 0, x_1^3 = 1, x_2^3 = 0, x_1^4 = 0, x_2^4 = 1, x_1^5 = 0, x_2^5 = 0, x_3^1 = 1, x_3^2 = 0, x_3^3 = 0, x_3^4 = 0, x_3^5 = 1)$, qui vaut 1294. Nous supprimons P^{011}, car il fournit une solution entière qui n'améliore pas la borne supérieure. Ainsi, nous avons $T = \{P^{0000}, P^{0001}\}$.

Nous choisissons P^{0000}, nous le relaxons puis le résolvons avec CPLEX. Ensuite, nous obtenons $(x_1^1 = 0.875, x_2^1 = 0.125, x_1^2 = 0, x_2^2 = 0, x_1^3 = 0, x_2^3 = 0.625, x_1^4 = 0.25, x_2^4 = 0.75, x_1^5 = 0.375, x_2^5 = 0, x_3^1 = 0, x_3^2 = 1, x_3^3 = 0.375, x_3^4 = 0, x_3^5 = 0.625)$, qui vaut 1279.25. Cette solution ne viole aucune inégalité valide, donc nous branchons sur x_2^3. Ainsi, nous obtenons deux sous-problèmes : $P^{00000} = P^{0000} \cup \{x_2^3 = 0\}$ et $P^{00001} = P^{0000} \cup \{x_2^3 = 1\}$. Après cela, nous mettons à jour la liste des nœuds actifs, $T = \{P^{0001}, P^{00000}, P^{00001}\}$.

Nous choisissons P^{0001}, le relaxons, puis le résolvons avec CPLEX. Après cela, nous obtenons $(x_1^1 = 1, x_2^1 = 0, x_1^2 = 0, x_2^2 = 1, x_1^3 = 0, x_2^3 = 1, x_1^4 = 0, x_2^4 = 0, x_1^5 = 0, x_2^5 =$

$0, x_3^1 = 0, x_3^2 = 0, x_3^3 = 0, x_3^4 = 1, x_3^5 = 1$), qui vaut 1260. Cette solution est entière et ne viole aucune inégalité valide. Donc nous mettons à jour la borne supérieure : **UB**=1260. Ensuite, nous supprimons P^{0001}, P^{00000}, et P^{00001} de T. Notons que P^{00000} et P^{00001} ne sont pas explorés car leur borne inférieure (1279.25) est supérieure à la borne supérieure (1260). C'est la fin de l'algorithme, car $T = \emptyset$.

L'arbre de recherche correspondant à cet exemple est visible dans la figure 29. Les nœuds en rouge sont ceux qui sont élagués, alors que ceux en vert ont permis de mettre à jour la borne supérieure (en d'autres termes, ceux qui ont donné les meilleures solutions entières). L'ordre par lequel les nœuds sont explorés est mis en évidence par les écritures en bleu (c'est un parcours en largeur d'abord).

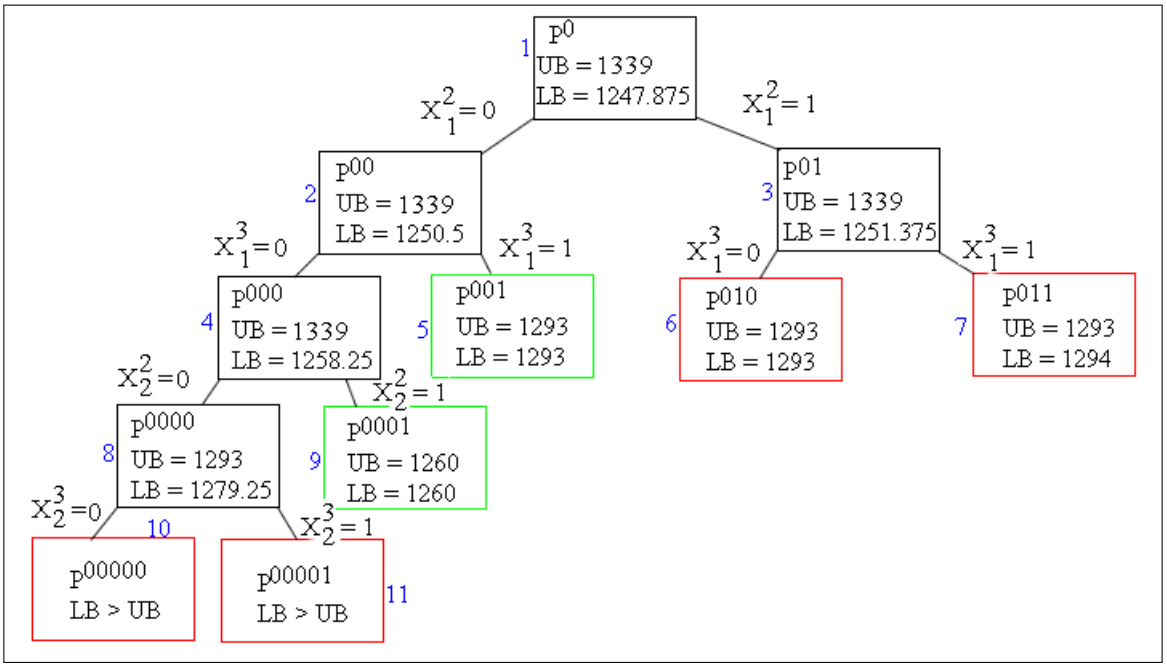


FIGURE 29 – Arbre de recherche.

4.5.3 Résultats numériques

La première étape de nos simulations numériques consiste à comparer le modèle mathématique que nous proposons à celui qui était proposé dans [79] par Moussi, et qui est formulé comme suit.

$$Minimiser \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} \sum_{k=1}^{N} x_{p,i}^k d_p^k \tag{4.1}$$

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} x_{p,i}^k = 1, \ \forall\, k = 1, ..., N \tag{4.2}$$

$$x_{p,i}^k + x_{p,i}^{k'} \leq 1, \ \forall\, k = 1, ..., N, \ k' = 1, ..., N, \ k \neq k', \\ \forall\, p = 1, ..., N_p, \ i = 1, ..., c_p \tag{4.3}$$

$$\sum_{k=1}^{N} x_{p,i}^{k} \geq \sum_{k=1}^{N} x_{p,i+1}^{k},\ \forall\, p = 1, ..., N_p,\ i = 1, ..., c_p - 1 \tag{4.4}$$

$$(r_p - R_k) x_{p,i}^{k} = 0,\ \forall\, k = 1, ..., N,\ p = 1, ..., N_p, i = 1, ..., c_p \tag{4.5}$$

$$T_k \leq M(1 - x_{p,i}^{k}) + x_{p,i}^{k} t_p,\ \forall\, k = 1, ..., N,\ p = 1, ..., N_p,\ i = 1, ..., c_p \tag{4.6}$$

$$\begin{aligned} M(1 - x_{p,i}^{k}) + T_k x_{p,i}^{k} \geq T_{k'} x_{p,i+1}^{k'},\ \forall\, k = 1, ..., N,\ k' = 1, ..., N,\ k' \neq k, \\ \forall\, p = 1, ..., N_p,\ i = 1, ..., c_p - 1 \end{aligned} \tag{4.7}$$

$$x_{p,i}^{k} \in \{0, 1\},\ \forall\, k = 1, ..., N,\ p = 1, ..., N_p,\ i = 1, ..., c_p \tag{4.8}$$

Ce modèle mathématique, que l'on nomme (F_1), peut être amélioré en modifiant la formulation de ses contraintes, afin de le rendre moins dur à exécuter avec CPLEX. On obtient ainsi, le modèle (F_2) suivant.

$$\min \sum_{k=1}^{N} \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} d_p^k x_{p,i}^{k} \tag{4.9}$$

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} x_{p,i}^{k} = 1,\ \forall\, k = 1, ..., N \tag{4.10}$$

$$\sum_{k=1}^{N} x_{p,i}^{k} \leq 1,\ \forall\, p = 1, ..., N_p,\ i = 1, ..., c_p \tag{4.11}$$

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} x_{p,i}^{k} = 0,\ \forall\, k = 1, ..., N \text{ tel que } R_k \neq r_p \text{ ou } T_k > t_p \tag{4.12}$$

$$\sum_{k=1}^{N} T_k x_{p,i}^{k} \geq \sum_{k=1}^{N} T_k x_{p,i+1}^{k}, \quad \forall\, p = 1, ..., N_p;\ i = 1, ..., c_p - 1 \tag{4.13}$$

$$x_{p,i}^{k} \in \{0, 1\}, \forall\, p = 1, ..., N_p,\ i = 1, ..., c_p,\ k = 1, ..., N \tag{4.14}$$

La formulation (4.11) est une amélioration de (4.3), qui a permis de diminuer $[N(N-1) - 1] \sum_{p=1}^{N_p} c_p$ contraintes. De même, la formulation (4.12) regroupe (4.5) et (4.6). En plus, elle prédétermine les variables de décision qui doivent inévitablement être nulles, ce qui permet de diminuer $2N \sum_{p=1}^{N_p} c_p$ contraintes. La formulation (4.13) allège (4.7), et rend inutile (4.4), ce qui permet de diminuer $N(N-1) \sum_{p=1}^{N_p} (c_p - 1) - \sum_{p=1}^{N_p} c_p$ contraintes.

Ces améliorations permettent de réduire la taille de mémoire nécessaire pour exécuter le modèle mathématique avec CPLEX. Par conséquent, elles permettent aussi de résoudre de plus grandes instances.

En ajoutant l'équation (4.15) suivante dans la formulation (F_2), on obtient la version sans remaniements du modèle décrit dans la section 4.3, que l'on nome (F_3).

$$\sum_{k=1}^{N}(N-k+1)x_{p,i}^{k} \geq \sum_{k=1}^{N}(N-k+1)x_{p,i+1}^{k} \quad \forall\, p=1,...,N_p,\ i=1,...,c_p-1 \tag{4.15}$$

Pour illustrer l'utilité des améliorations effectuées sur le modèle (F_1), nous avons fait des simulations numériques avec CPLEX 12.6. Pour ce faire, nous avons utilisé **Concert Technology** pour coder en langage C++ deux programmes informatiques qui sont différenciés par le modèle mathématique considéré (F_1 ou F_2). Les instances utilisées, dans cette phase de simulation, sont pareilles que celles utilisées par Moussi dans [79]. Ce dernier avait considéré deux groupes d'instances (instances de tailles moyennes et instances de grandes tailles), qui sont décrites dans le tableau 4.7.

Tableau 4.7: Description des instances

	Petite taille	Grande taille
Tailles de conteneur considérées	20 pieds, 40 pieds, et 45 pieds	
Capacité maximale d'une pile	3 conteneurs	
Date de départ d'un conteneur à stocker	$10h \leq T_k \leq 80h$	$200h \leq T_k \leq 1000h$
Date de départ d'un conteneur qui est déjà au sommet d'une pile	$40h \leq t_p \leq 100h$	$500h \leq t_p \leq 2000h$
Nombre de conteneurs	$10 \leq N \leq 95$	$N \geq 100$
Nombre de piles	$20 \leq N_p \leq 110$	$N_p \geq 3000$

Les tableaux 4.8 et 4.9 montrent respectivement les résultats numériques obtenus avec CPLEX sur de petites et de grandes instances, en considérant les modèles mathématiques (F_1) et (F_2). Dans ces tableaux, N représente le nombre de conteneurs à stocker, N_p est le nombre de piles dans la cour de stockage, et P_d est le pourcentage d'emplacements de stockage libres ($P_d = \frac{Somme\_des\_positions\_libres}{3\times N_p} \times 100$).

Tableau 4.8: Résultats de CPLEX pour les modèles (F_1) et (F_2) avec des instances de tailles moyennes

Type d'instance	Description			(F_1)		(F_2)	
	N_p	N	P_d	Résultat	Durée	Résultat	Durée
	20	10	80%	4150	0.02 sec	4150	0.01 sec
	35	25	67.61%	15350	0.06 sec	15350	0.04 sec
	40	20	68.33%	10500	0.04 sec	10500	0.04 sec

Instances de tailles moyennes		25	73.33%	17650	0.09 sec	17650	0.02 sec
	45	30	57.03%	22750	0.09 sec	22750	0.03 sec
		35	66.66%	21250	0.17 sec	21250	0.03 sec
	50	30	60%	21700	0.1 sec	21700	0.03 sec
		35	66.66%	31500	0.14 sec	31500	0.03 sec
		40	71.33%	27750	0.28 sec	27750	0.05 sec
	55	40	66.66%	31100	0.24 sec	31100	0.02 sec
	60	40	65%	32200	0.22 sec	32200	0.04 sec
		45	64.44%	43250	0.29 sec	43250	0.03 sec
	65	40	64.61%	33850	0.3 sec	33850	0.03 sec
	70	45	67.14%	44200	0.37 sec	44200	0.04 sec
		50	67.14%	53150	0.53 sec	53150	0.05 sec
		55	73.33%	56050	0.7 sec	56050	0.06 sec
		60	61.42%	72850	0.74 sec	72850	0.06 sec
	75	60	67.11%	66200	0.87 sec	66200	0.1 sec
	80	60	65%	60850	0.93 sec	60850	0.1 sec
		65	68.33%	84400	1.13 sec	84400	0.1 sec
	85	60	68,62%	60950	1.07 sec	60950	0.07 sec
	90	65	69.62%	64400	1.27 sec	64400	0.09 sec
		70	64.44%	87150	1.3 sec	87150	0.1 sec
	95	70	70.17%	78300	1.55 sec	78300	0.14 sec
		75	66.31%	91750	1.82 sec	91750	0.15 sec
		80	61.4%	112000	1.98 sec	112000	0.11 sec
	100	80	64%	122600	2.12 sec	122600	0.11 sec
		85	63%	111300	2.33 sec	111300	0.11 sec
	105	95	69.2%	132750	3.36 sec	132750	0.14 sec
	110	95	66.96%	145600	3.37 sec	145600	0.13 sec
		90	65.75%	124100	2.98 sec	124100	0.2 sec

Tableau 4.9: Résultats de CPLEX pour les modèles (F_1) et (F_2) avec des instances de grandes tailles

Type	Description			(F_1)		(F_2)	
d'instance	N_p	N	P_d	Résultat	Temps	Résultat	Temps
Instances de grandes tailles	300	100	64.22%	123717	25.62 sec	123717	0.35 sec
		200	66.66%	Mémoire insuffisante		514861	1.17 sec
		300	66.66%			1190360	1.48 sec
	400	100	67.08%			150262	0.46 sec
		200	66.5%			527565	1.39 sec
		300	50.16%			1082214	2.65 sec
		400	66.33%			2091114	3.62 sec
		500	66.33%			3265898	10.19 sec
	500	100	66.53%			113174	0.56 sec
		200	67.13%			515642	1.84 sec
		300	65.93%			1221327	3.69 sec
		400	67.2%			2131653	7.15 sec

600	100	66.66%		124989	0.74 sec
	200	66.72%		527035	2.17 sec
	300	67.94%		1205836	3.84 sec
	400	67.77%		2067877	8.06 sec
700	100	67.42%		102357	0.9 sec
	200	67.52%		148935	1.69 sec
	300	67.14%		1185032	3.19 sec
	400	66.61%		2009249	6.65 sec
	500	66.09%		3131746	10.5 sec
	600	50%		4194030	11.87 sec
1500	1200	50%		19838470	158.6 sec
	1300	49%		Mémoire insuffisante	

Dans les tableaux 4.8 et 4.9 on peut remarquer que les modèles mathématiques (F_1) et (F_2) donnent les mêmes solutions. Cependant, l'utilisation du modèle (F_2) est plus avantageux, car il réduit les durées d'exécution. On peut aussi remarquer que (F_2) permet à CPLEX de résoudre de plus grandes instances, qu'il ne peut résoudre en utilisant (F_1). Pour chacun de ces deux modèles, les durées d'exécution avec CPLEX ne sont pas coûteuses, mais on est confronté à un problème d'espace mémoire, qui fait échouer l'exécution de certaines instances. Ce phénomène s'accentue en fonction du nombre d'inégalités ou d'égalités contenus dans le modèle mathématique considéré. Ainsi, on peut affirmer que l'ajout de contraintes opérationnelles (telles que l'ordre de déchargement des conteneurs) dans le modèle aura pour effet de rendre son exécution avec CPLEX plus dur. Raison pour laquelle nous proposons l'algorithme de branch-and-cut (nommé PSC-BC), décrit dans la section 4.5, pour résoudre le cas du stockage où l'on tient compte de l'ordre d'arrivée des conteneurs. Nous avons codé cet algorithme de branch-and-cut en langage C++ en utilisant SCIP. C'est un logiciel très adapté à l'implémentation d'un algorithme de branch-and-cut, car il est conçu de telle sorte que l'utilisateur puisse avoir un contrôle total sur les processus de résolution. Les résultats de l'algorithme de branch-and-cut que nous proposons sont notés dans le tableau 4.19.

Tableau 4.10: Résultats de l'algorithme de Branch-and-cut (PSC-BC)

Type d'instance	Description			CPLEX		PSC-BC	
	N_p	N	P_d	Résultat	Temps	Résultat	Temps
Instances de grandes tailles	200	150	65.77 %	55011	53 min 50 sec	55011	1 sec
	3500	100	54.87 %	Mémoire insuffisante		32547	0 sec
		200	69 %			66300	3 sec
		300	63.39 %			90979	14 sec
		400	56.69 %			123438	41 sec
		500	57.43 %			168895	1 min 36 sec
		600	53 %			183415	6 min 13 sec
		700	56.28 %			215138	5 min 57 sec
		800	68.63 %			243917	9 min 49 sec
		900	58.78 %			274238	15 min 35 sec
		1000	56.75 %			320415	1 h 41 min 26 sec
		1100	55.47 %			337347	1 h 43 min 47 sec
		1200	66.03 %			373498	2 h 5 min 4 sec
		1300	59.11 %			403054	2 h 21 min 20 sec
		1400	65.36 %			...	

Le symbole ... signifie que l'exécution est interrompue volontairement car elle a dépassé 3 heures de temps.

Dans le tableau 4.19, les résultats de CPLEX sont obtenus en considérant le modèle mathématique de coloration bornée décrit dans la section 4.5. La première ligne du tableau montre que l'algorithme PSC-BC (que nous avons codé avec SCIP) permet de réduire les temps de calcul. Outre ce constat, les résultats numériques montrent que le PSC-BC permet d'améliorer la limite de CPLEX, en résolvant des instances qui n'ont pas pu être résolues directement avec CPLEX (que ça soit avec le modèle de coloration bornée ou bien avec le modèle (F_3)).

4.6 Algorithme de colonie d'abeilles (PSC-ACA)

Dans le cas d'un terminal à conteneurs qui ne dispose pas d'une cour de stockage assez grande pour permettre d'effectuer les opérations de stockage et d'extraction sans aucun remaniement, nous proposons un algorithme de colonie d'abeilles qui minimise simultanément le nombre de remaniements, les distances à parcourir, et les dispersions de conteneurs qui appartiennent une même catégorie.

L'algorithme de colonie d'abeilles est un algorithme méta-heuristique qui est applicable à divers problèmes d'optimisation. Il fut inventé par Pham et al., qui ont publié en 2005 "*The bees algorithm - A novel tool for complex optimization problems*" [85]. Ces derniers furent inspirés par le comportement des abeilles naturelles quand elles recherchent des fleurs pour la préparation de miel. Des informations concernant ces animaux sont données dans la section 4.6.1. Après cela, l'algorithme de colonie d'abeilles que nous proposons pour la résolution du problème de stockage de conteneurs sera détaillé dans la

section 4.6.2. Les résultats numériques obtenus avec cet algorithme seront présentés dans la section 4.6.4.

4.6.1 Comportement des abeilles dans la nature

Des études sur les colonies d'abeilles ont permis de mieux comprendre ces animaux. Parmi ces travaux, on peut citer ceux de Von [106] et Seeley [98], qui ont révélé qu'une colonie d'abeilles mellifères est capable de parcourir simultanément de longues distances (plus de 10 km) et dans plusieurs directions pour explorer un grand nombre de sources de nourriture. D'autres travaux, en l'occurrence ceux de Bonabeau [11] et de Camazine [28], ont mis en évidence que dans une colonie d'abeilles les tâches sont réparties et que ce sont les butineuses qui sont chargées d'exploiter les endroits où il y a des fleurs. En principe, plus d'abeilles sont déployées dans les localités où la récolte nécessite moins d'effort, c'est-à-dire les endroits où il y a des quantités abondantes de nectar ou bien de pollen. Avant le début d'une récolte, des éclaireuses sont déployées pour rechercher des champs de fleurs. Lorsque ces dernières font des découvertes, elles essayent de trouver de meilleures, avant de retourner dans leur ruche pour informer les butineuses. Selon Seeley [98], même pendant les périodes de récolte, les éclaireuses continuent à chercher de nouvelles sources de fleurs. Lorsque des éclaireuses reviennent dans leur ruche, suite à une découverte de fleurs qui ont une bonne qualité (mesurée par une combinaison de certains composants, tel que la teneur en sucre), elles déposent leur récolte et vont sur la "piste de danse" pour effectuer la "danse frétillante" [106]. Cette danse mystérieuse est le moyen de communication des abeilles. Elle permet de renseigner trois types d'informations concernant un site de fleurs : sa direction, sa distance par rapport à la ruche, et sa qualité [106, 28]. Grâce à ces informations, les butineuses pourront retrouver avec précision l'endroit où se trouve le site. Chaque information sur le milieu extérieur est recueillie individuellement à partir de la danse frétillante. Ainsi, selon Camazine [28], cette danse permet à la colonie d'évaluer les mérites de différents sites de fleurs en fonction de leurs qualités et des efforts nécessaires pour les exploiter. Après la danse frétillante, les éclaireuses retournent à leurs tâches, ensuite des nombres inégaux de butineuses sont envoyés sur les champs renseignés. Cependant, plus d'abeilles sont envoyées sur les champs les plus prometteurs, cela permet à la colonie d'effectuer une récolte rapide et efficace. Pendant l'exploitation d'un site, la quantité de ressource est continuellement surveillée. Ceci, afin d'effectuer une nouvelle danse frétillante pour solliciter plus de butineuses si la quantité de nourriture reste remarquable.

4.6.2 Quelques notions utiles

L'algorithme de colonie d'abeilles que nous proposons pour résoudre le problème de stockage de conteneurs a pour objectif la minimisation simultanée de trois éléments : la distance totale à parcourir entre les piles et les portes de sortie des conteneurs, le nombre total de remaniements, et les écarts entre les emplacements de stockage des conteneurs qui appartiennent à une même catégorie. On a donc un problème d'optimisation multi-objectif qui, selon Akbari et al. [3], peut être résolu par cinq approches différentes : la méthode lexicographique, la technique de sous-population, l'approche basée sur les notions de Pareto, la méthode d'agrégation pondérée, et les méthodes hybrides.

- La technique lexicographique consiste à classer les fonctions objectif suivant leurs importances, et ensuite minimiser chacune d'elle séparément afin d'obtenir la solution optimale [23, 53].
- Une méthode de sous-population divise une population donnée en plusieurs sous-populations. Chacune de ces dernières est destinée à l'optimisation de l'une des fonctions objectifs. Cependant, un compromis entre les différentes solutions est généralement obtenu grâce à des échanges d'informations entre les sous-populations [83, 27].
- Les méthodes basées sur les notions de Pareto utilisent la "*dominance au sens de Pareto*" [69, 68], qui est définie comme suit. Soit X un espace de recherche n-dimensionnel et $f_i,\ i = 1, ..., k$, k fonctions objectifs définies sur X. En supposant qu'on ait ces m contraintes suivantes

  $$g_i \leq 0, \quad i = 1, ..., m,$$

  un problème d'optimisation multi-objectif consiste à trouver un vecteur $x^* = (x_1^*, x_2^*, ..., x_n^*) \in X$ qui satisfait les contraintes et qui optimise le vecteur fonction objectif $f(x) = [f_1(x), f_2(x), ..., f_k(x)]$. Les fonctions objectifs peuvent être en conflit. Ainsi, dans plusieurs cas, il est impossible de trouver une solution qui les optimise toutes. Dans ces cas là, on a tendance à chercher l'ensemble des solutions qui sont optimales au sens de Pareto. Considérons deux vecteurs $u = (u_1, ..., u_k)$ et $v = (v_1, ..., v_k)$. Dans le cas d'un problème de minimisation, on dit que u *domine* v si et seulement si $u_i \leq v_i, \forall\ i = 1, ..., k$ et $u_i < v_i$ pour au moins une composante. Dans le cas d'un problème de maximisation, on remplace respectivement $\leq$ et $<$ par $\geq$ et $>$, dans la définition précédente. Cette propriété est appelée "*dominance au sens de Pareto*", elle est utilisée pour déterminer les solutions qui sont optimales au sens de Pareto. Ainsi, une solution x d'un problème d'optimisation multi-objectif est "*optimale au sens de Pareto*", si et seulement si, il n'existe pas une autre solution y qui est tel que $f(y)$ domine $f(x)$. L'ensemble de toutes les solutions qui sont optimales au sens de Pareto est appelé "*l'ensemble Pareto*", notons le P^*. Le "*front Pareto*" est l'ensemble $PF^* = \{(f_1(x), f_2(x), ..., f_k(x)) | x \in P^*\}$. Un front Pareto est convexe si et seulement si $\forall\ u, v \in PF^*,\ \forall \lambda \in (0, 1), \exists w \in PF^* : \lambda||u|| + (1 - \lambda)||v|| \geq ||w||$. Similairement, un front Pareto est concave si et seulement si $\forall\ u, v \in PF^*,\ \forall \lambda \in (0, 1), \exists w \in PF^* : \lambda||u|| + (1 - \lambda)||v|| \leq ||w||$. Un front Pareto peut aussi être partiellement convexe, ou partiellement concave, ou discontinu.
- L'approche d'agrégation pondérée consiste à combiner toutes les fonctions objectifs f_i, pour obtenir une seule fonction objectif qui est de la forme $F = \sum_{i=1}^{k} w_i f_i(x)$, où les $w_i,\ i = 1, ..., k$, sont des nombres réels positifs appelés poids. Généralement on a $\sum_{i=1}^{k} w_i = 1$. Ces poids peuvent être fixés ou bien dynamiquement ajustés au cours des processus d'optimisation [86].

  Si les poids sont fixés, alors on est dans le cas d'une "*agrégation pondérée conventionnelle*". Cette approche nécessite une bonne connaissance de l'espace de recherche afin de choisir les poids appropriés. En plus de cela, elle fournit une unique solution optimale au sens de Pareto, à chaque exécution. Cependant, pour obtenir plusieurs solutions optimales au sens de Pareto, il suffit de faire plusieurs exécutions.

  Dans le cas d'un problème d'optimisation qui a deux fonctions objectifs, les poids peuvent être modifiés durant le processus d'optimisation, comme le décrit ces

équations suivantes.

$$w_1(t) = sign(sin(2\pi t/F)), \quad w_2(t) = 1 - w_1(t)$$

où t est le numéro de l'itération courante, F est la fréquence de changement des poids, et $sign$ est la fonction définie comme suit

$$sign(x) = \begin{cases} -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ 1 & \text{si } x > 0 \end{cases}$$

C'est ce que l'on appelle l'approche "*d'agrégation pondérée Bang-Bang*" [86], dans laquelle les poids changent de façon brusque grâce à l'utilisation de la fonction $sign(.)$. Alternativement, les poids peuvent être modifiés graduellement en utilisant les équations suivantes

$$w_1(t) = |sin(2\pi t/F)|, \; w_2(t) = 1 - w_1(t).$$

Cette technique est appelée "*agrégation pondérée dynamique*".

- L'approche hybride consiste à combiner deux ou plusieurs des quatre méthodes décrites ci-dessus, afin d'exploiter leurs avantages [67, 113].

4.6.3 Description de l'algorithme

Nous utilisons une approche d'agrégation pondérée conventionnelle pour résoudre le problème de stockage de conteneurs décrit dans la section 4.3. Pour ce faire, nous remplaçons la fonction objectif

$$\left(\sum_{k=1}^{N} \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} d_p^k x_{p,i}^k, \sum_{k=1}^{N} \sum_{p=1}^{N_p} y_p^k, \sum_{k=1}^{N-1} \sum_{k'=k+1}^{N} z_k^{k'} \right)$$

par

$$\sum_{k=1}^{N} \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} d_p^k x_{p,i}^k + \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} M y_p^k + \sum_{k=1}^{N-1} \sum_{k'=k+1}^{N} z_k^{k'}$$

où M est un nombre entier très grand.

Cette formulation montre l'importance d'éviter les remaniements. Certes la minimisation des distances et le regroupement des conteneurs qui appartiennent à une même catégorie peuvent contribuer à accélérer les opérations de chargement (de navires, trains, ou camions). Cependant, ils ne serviraient pas à grand chose, si le nombre de remaniements est énorme. Cela se justifie par le fait qu'un seul remaniement peut nécessiter plusieurs allers-retours d'un cavalier, afin de déplacer les conteneurs qui constituent des obstacles et d'accéder au conteneur désiré. En plus de cela, s'ajoute la nécessité de trouver des emplacements convenables où stocker les conteneurs déplacés sans provoquer de futurs remaniements, avant de les replacer dans leurs piles d'origines.

L'algorithme de colonie d'abeilles (PSC-ACA), que nous proposons pour la résolution du problème de stockage de conteneurs, fait intervenir six paramètres : le nombre maximum d'itérations (NI_{max}), le nombre d'éclaireuses parmi les abeilles (F_s), le nombre d'élites (N_e) à sélectionner parmi les sites de fleurs trouvés par les éclaireuses, le nombre (N_b) de sites à sélectionner parmi les meilleurs sites qui suivent les élites, le nombre (F_e) de butineuses à envoyer dans les sites élites, le nombre (F_b) de butineuses à envoyer dans les autres sites sélectionnés.

La figure 30 est une représentation de l'algorithme PSC-ACA.

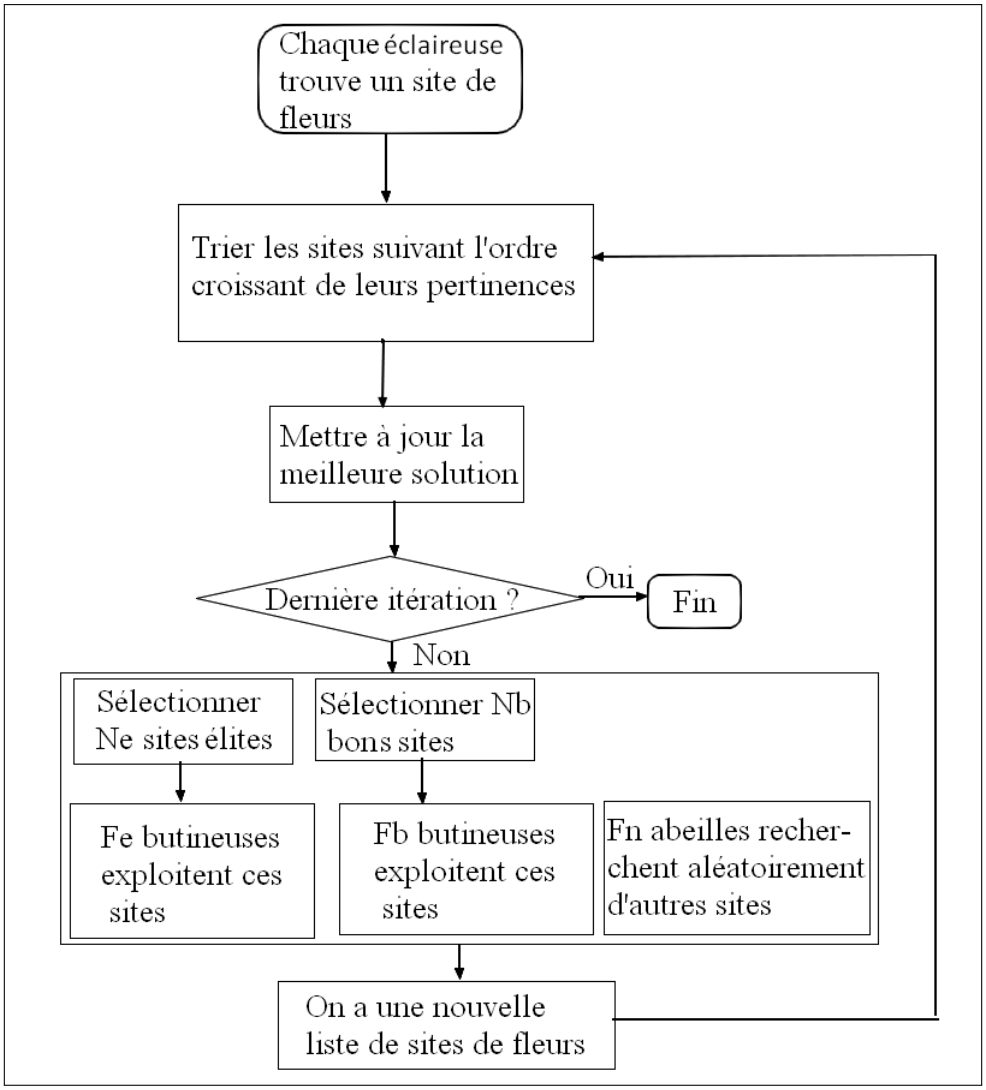


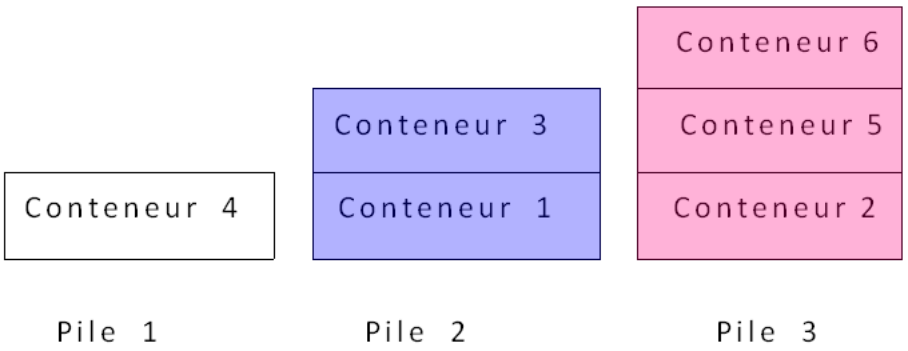
FIGURE 30 – Algorithme PSC-ACA

Codage d'une solution

Dans l'algorithme PSC-ACA, chaque site désigne une solution. Nous représentons chaque solution sous forme d'un tableau qui a deux lignes et dont le nombre de colonnes est égal au nombre de conteneurs à stocker. Dans la première ligne, sont écrits les numéros des conteneurs, alors que les numéros des piles sélectionnées sont écrits dans la deuxième ligne. L'exemple suivant est une illustration d'un stockage de six conteneurs dans trois piles.

Conteneur →	4	2	1	3	5	6
Pile →	1	3	2	2	3	3

Cette solution correspond à la disposition suivante.



Si plusieurs conteneurs sont affectés à une même pile, alors ils seront stockés suivant l'ordre croissant de leurs numéros de colonnes. Cela permet de tenir compte des contraintes d'ordres d'arrivée (4.5) et de départ (4.8). Par exemple, les conteneurs 1 et 3 sont affectés à la pile 2, mais le conteneur 1 est stocké en dessous du conteneur 3.

Méthode de recherche d'un site

La méthode de recherche d'un site correspond à la méthode de construction d'une solution, puisque dans notre algorithme de colonie d'abeilles un site correspond à une solution du problème de stockage de conteneurs.

Pour construire une solution, on affecte les conteneurs un par un dans un ordre aléatoire, comme le montre l'algorithme suivant.

1. $a = 1$
2. $S = (s_{i,j})_{\substack{1 \leq i \leq 2 \\ 1 \leq j \leq N}}$ (c'est la solution que l'on est entrain de créer)
3. Tant que $(a \leq N)$ faire
   - 3.a. Choisir aléatoirement un conteneur k qui n'est pas encore traité
   - 3.b. Choisir aléatoirement une pile p qui n'est pas pleine, qui a la même taille que k, et qui respecte les contraintes d'ordres d'arrivée et de départ, c'est-à-dire qui ne contient pas de conteneur k' qui est tel que $k' > k$ ou $T_{k'} < T_k$
   - 3.c. $s_{1,a} = k$, et $s_{2,a} = p$
   - 3.d. $a = a + 1$
4. Fin tant que

Méthode d'exploration des sites

Nous utilisons une méthode de recherche locale pour explorer les sites, c'est-à-dire améliorer les solutions. Pour ce faire, nous choisissons aléatoirement un conteneur, ensuite nous l'affectons à une autre pile. De préférence, la nouvelle pile que l'on assigne à un conteneur doit être mieux que la précédente. En d'autres termes, elle doit minimiser la fonction objectif décrite dans la section 4.6.3. L'algorithme de recherche de solution voisine est la suivante.

1. Fin = faux ;
2. Compteur = 1 ;
3. Tant que (Fin = faux, et Compteur $\leq N$) faire
    3.a. Sélectionner aléatoirement un conteneur k qui n'est pas encore traité
    3.b. Soit p la pille affectée au conteneur k dans la solution courante
    3.c. Construire la liste L des piles qui ne sont pas pleines, qui ont la même taille que k, et qui vérifient les contraintes d'ordres
    3.d. Changement = faux
    3.e. j=1
    3.f. Tant que (Changement = faux, et $j \leq card(L)$) faire
        3.f.1. $p' = L(j)$
        3.f.2. Si $(d^k_{p'} + My^k_{p'} + A < d^k_p + My^k_p + B)$ alors
            3.f.2.a. Changement = vrai
        3.f.3. Sinon
            3.f.3.a. $j = j + 1$
    3.g. Fin tant que
    3.h. Si (Changement = vrai) alors
        3.h.a. Affecter k à p'
        3.h.b. Mettre à jour les capacités des piles p et p'
        3.h.b. Fin = vrai
    3.i. Sinon
    3.j. Compteur = Compteur +1
4. Fin tant que

$A = \sum_{k'>k}^{N} z^{k'}_k$, sachant que le conteneur k est affecté à la pile p.
$B = \sum_{k'>k}^{N} z^{k'}_k$, sachant que le conteneur k est affecté à la pile p'.

Pseudo-code de l'algorithme de colonie d'abeilles

L'algorithme CA-PCS progresse comme suit.

1. Chaque éclaireuse trouve une solution
2. Évaluer chaque solution
3. Initialiser le nombre d'itérations, i=1
4. Tant que $(i < NI_{max})$ faire
    4.a. Sélectionner N_e solutions élites
    4.b. Sélectionner N_b solutions parmi les meilleures
    4.c. F_e butineuses améliorent les solutions élites
    4.d. Évaluer les solutions améliorées, puis sauvegarder les N_e meilleures parmi elles
    4.e. F_b butineuses améliorent les autres solutions sélectionnées à l'étape 4.b
    4.f. Évaluer les solutions améliorées, puis sauvegarder les N_b meilleures parmi elles
    4.g. Les abeilles restantes construisent aléatoirement de nouvelles solutions

4.h. i=i+1
5. Fin tant que

4.6.4 Résultats numériques

Puisque nous considérons, dans cette section, le cas d'un terminal à conteneurs qui ne dispose pas, à chaque instant, de suffisamment de piles vides pour que l'élimination complète des remaniements soit possible, nous n'utilisons donc pas les instances considérées dans [79]. Ainsi, nous générons aléatoirement de nouvelles instances qui ne peuvent pas être résolues sans la tolérance de remaniements. On peut comprendre par là que le nombre de piles vides, de chacune de ces instances, est strictement inférieur au nombre de conteneurs à stocker. En plus de cela, contrairement au cas du stockage sans aucun remaniement (voir la section précédente) où les dates de départ des conteneurs qui sont déjà aux sommets des piles, au début des opérations de stockage, sont toujours supérieures à celles des nouveaux conteneurs à stocker, dans cette section, nous utilisons des instances qui sont générées de telle sorte que les dates de départ des conteneurs qui sont aux sommets des piles au début des opérations de stockage (de la période de travail courante) ne sont pas toujours supérieures à celles des nouveaux conteneurs à stocker. Ce procédé est plus adapté à la réalité, car vu que les autorités portuaires ont tendance à imposer que la durée de séjour de chaque conteneur soit inférieure à une limite fixée [63], la probabilité que la date de départ d'un ancien conteneur soit plus petite que celle d'un nouveau conteneur est plus grande que le contraire. Les informations concernant des données utilisées dans les simulations numériques effectuées pour la résolution du cas du stockage avec remaniement sont notées dans le tableau 4.14.

Tableau 4.14: Description des instances

Description	Valeur
Nombre de conteneurs	$10 \leq N \leq 105$
Nombre de piles	$18 \leq N_p \leq 95$
Pourcentage d'emplacements inoccupés	$45\% \leq \frac{100 \times \sum_{p=1}^{N_p} c_p}{3 \times N_p} \leq 65\%$
Date de départ d'un nouveau conteneur à stocker	$10 \text{ h} \leq T_k \leq 80 \text{ h}$
Date de départ d'un conteneur qui est déjà au sommet d'une pile	$5 \text{ h} \leq t_p \leq 40 \text{ h}$
Taille d'un conteneur ou bien d'une pile	$R_k,\ r_p \in \{20 \text{ pieds}, 40 \text{ pieds}, 45 \text{ pieds }\}$

Fixation des paramètres de l'algorithme de colonie d'abeilles

La première étape des simulations numériques consiste à déterminer les bonnes valeurs des paramètres de l'algorithme de colonie d'abeilles que nous proposons pour la résolution du problème de stockage de conteneurs. C'est une phase capitale, car ces dernières ont

une influence sur la qualité des solutions fournies par l'algorithme. Aucun paramètre n'est négligé, puisqu'ils sont tous importants. Cependant, le nombre d'itérations et le nombre de solutions initiales à chaque itération sont des éléments clés, parce qu'ils agissent aussi sur la durée d'exécution de l'algorithme. Ainsi, nous commençons par fixer le nombre d'itérations, ensuite le nombre de solutions initiales (c'est-à-dire le nombre d'abeilles éclaireuses, puisque chacune d'elles fournit une solution initiale), avant de traiter les autres paramètres de l'algorithme. Le nombre d'itérations ne doit pas être trop petit, ni trop grand, car dans le premier cas les solutions risquent de ne pas être de bonne qualité, alors que dans le second cas, les durées d'exécution risquent d'être trop longues. Ainsi, pour connaître sa valeur appropriée, nous fixons les autres paramètres et le faisons varier entre 20 et 100 par pas de 10. Nous testons 130 instances et notons, pour chacune d'elles, le nombre d'itérations à partir duquel la valeur de la fonction objectif ne diminue plus. Cela signifie que si l'optimum d'une instance est trouvé avec 20 itérations et aussi avec 30 itérations, alors on considère que cette instance ne nécessite pas de plus de 20 itérations. Pour chaque instance, on considère uniquement le plus petit nombre d'itérations qui donne la meilleure solution. On compte le nombre d'instances retenues pour chaque nombre d'itérations, ensuite on reporte les résultats dans le tableau 4.15.

Tableau 4.15: Fixation du nombre d'itérations

Nombre d'itérations	Nombre de meilleurs résultats
20	76
30	11
40	6
50	10
60	6
70	7
80	4
90	5
100	5
Moyenne pondérée	36.23

La moyenne pondérée est calculée comme suit :

$$\frac{20 \times 76 + 30 \times 11 + 40 \times 6 + 50 \times 10 + 60 \times 6 + 70 \times 7 + 80 + 90 \times 5 + 100 \times 5}{130} = 36.23$$

On fixe alors le nombre d'itérations à 37.

Nous suivons la même procédure pour fixer le nombre de solutions initiales (d'abeilles éclaireuses).

Les résultats sont visibles dans le tableau 4.16.

Tableau 4.16: Fixation du nombre d'abeilles éclaireuses

Nombre d'abeilles éclaireuses	Nombre de meilleurs résultats
10	63
20	10
30	6
40	6
50	8
60	7
70	8
80	8
90	5
100	9
Moyenne pondérée	35.54

La moyenne pondérée est calculée comme suit

$$\frac{10 \times 63 + 20 \times 10 + 30 \times 6 + 40 \times 6 + 50 \times 8 + 60 \times 7 + 70 \times 8 + 80 \times 8 + 90 \times 5 + 100 \times 9}{130}$$

Après ces calculs, nous fixons à 36 le nombre d'abeilles éclaireuses.

Les autres paramètres sont aussi traités un par un. Le tableau 4.17 montre les valeurs qui leurs sont attribuées.

Tableau 4.17: Valeurs des paramètres de l'algorithme de colonie d'abeilles

Paramètre	Notation et valeur
Nombre d'itérations	$NI_{max} = 37$
Nombre d'abeilles éclaireuses	$F_s = 36$
Nombre de solutions élites	$N_e = 8$

Nombre de meilleures solutions après les élites	$N_b = 10$
Nombre de butineuses pour les solutions élites	$F_e = 27$
Nombre de butineuses pour les meilleures solutions	$F_b = 19$

Évaluation de l'algorithme de colonie d'abeilles

Pour tester la qualité des solutions fournies par l'algorithme de colonie d'abeilles, nous les comparons aux solutions optimales fournies par le logiciel "ILOG CPLEX". Les propriétés générales des instances utilisées dans cette section sont décrites dans le tableau 4.14. Cependant, plus de précisions sont données dans le tableau 4.18.

Tableau 4.18: Résultats numériques de l'algorithme de colonie d'abeilles. NR est le nombre de remaniements.

Instances			CPLEX				PSC-ACA			
N_p	N	P_d	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR
18	11	53%	59055	0.05 sec	9019	5	59057	1 sec	9019	5
36	27	49%	132448	7.06 sec	21807	11	142902	1 sec	22160	12
44	19	50%	65356	0.48 sec	14983	5	65712	2 sec	15276	5
	26	50%	81533	3.74 sec	20860	6	92435	2 sec	21372	7
47	30	48%	95079	1.55 sec	22389	7	116717	3 sec	25667	9
	34	48%	168504	15.56 sec	26912	14	230735	1 sec	28851	20
51	28	49%	83940	6.25 sec	22940	6	116717	3 sec	25667	9
	36	49%	131034	40.09 sec	29345	10	142194	4 sec	30279	11
53	35	49%	120298	6 min 27 sec	28488	9	140818	5sec	29029	11
59	41	48%	145484	1 min 49 sec	33405	11	167666	5 sec	34612	13
	44	48%	168419	2 h 21 min 58 sec	35619	13	200238	6 sec	36626	16
65	45	52 %	Mémoire insuffisante				51329	17 sec	37668	1
68	46	51 %					71911	16 sec	38511	3
	49	51 %					125524	7 sec	41032	8
	57	63%					133705	14 sec	48063	8
	62	63%					219060	5 sec	52824	16
73	63	65%					262072	11 sec	61886	19
80	61	63%					139741	20 sec	51699	8
	66	63%					155627	23 sec	56467	9
91	68	62%					188283	27 sec	58240	12
	75	62%					178581	32 sec	64851	10
96	69	63%					171176	29 sec	58331	10
	74	63%					189947	33 sec	63889	11
	81	63%					195564	38 sec	69261	11
101	80	63%					217763	39 sec	68699	13
	85	63%					261347	41 sec	72556	17
104	94	64%					249761	50 sec	81360	14
110	94	65%					236662	54 sec	79160	13

	91	65%		157704	54 sec	76646	5

Les résultats du tableau 4.18 montrent que, lorsque l'on considère le cas du stockage où les remaniements sont inévitables et où l'on regroupe les conteneurs par catégorie, les durées d'exécution nécessaires au logiciel "ILOG CPLEX" augmentent ainsi que la taille de mémoire nécessaire. Cependant, on peut également constater que l'algorithme de colonie d'abeilles est assez rapide et qu'il donne de bonnes solutions. En plus de cela, il est capable de résoudre rapidement des instances qui n'ont pas pu être résolues par CPLEX. Une comparaison entre les distances trouvées par le logiciel "ILOG CPLEX" et l'algorithme PSC-ACA est donnée dans le tableau 4.19.

Tableau 4.19 – Comparaison entre PSC-ACA et CPLEX

Instances			Distance		Gap
N_p	N	P_d	PSC-ACA	CPLEX	
18	11	53%	9019	9019	0%
36	27	49%	22160	21807	1.61%
44	19	50%	15276	14983	1.95%
	26	50%	21372	20860	2.45%
47	30	48%	26912	22389	14.64%
	34	48%	28851	25667	7.20%
51	28	49%	25667	22940	11.89%
	36	49%	30279	29345	1.90%
53	35	49%	29029	28488	3.18 %
59	41	48%	34612	33405	3.61%
	44	48%	36626	35619	2.83%
			Moyenne		4.37%

Le pourcentage de déviation est calculé comme suit.

$$gap = \frac{Obj_{Algorithme} - Obj_{CPLEX}}{Obj_{CPLEX}} \times 100$$

où $Obj_{Algorithme}$ est la valeur de la meilleure solution trouvée par l'algorithme métaheuristique (ici l'algorithme de colonie d'abeilles), et Obj_{CPLEX} est la valeur de la solution optimale trouvée par le logiciel "ILOG CPLEX".

Comme on peut le constater dans le tableau 4.19, les distances trouvées par l'algorithme PCS-ACA ne sont pas loin de celles trouvées par CPLEX. D'ailleurs le pourcentage de déviation moyen ne dépasse pas 5%. Cet algorithme est aussi capable de minimiser le nombre de remaniements, comme l'illustre la figure 31

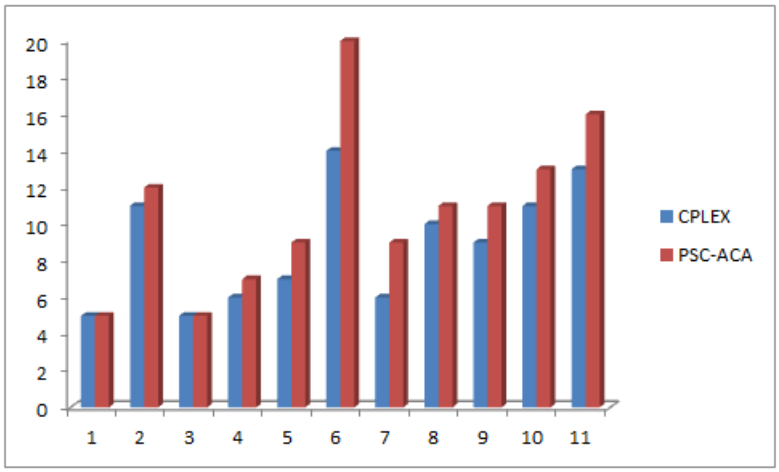


FIGURE 31 – Comparaison des nombres de remaniements

Sur la figure 31, on peut remarquer que dans certains cas, l'algorithme PSC-ACA et CPLEX trouvent les mêmes nombres de remaniements, et que dans la plupart des cas, leurs résultats sont assez proches.

4.7 Conclusion

Dans ce chapitre, nous avons traité le cas statique du problème de stockage de conteneurs dans un terminal portuaire. Une modélisation mathématique, dans laquelle on tient compte des contraintes physiques et opérationnelles du problème, est proposée. Trois principaux objectifs sont visés dans ce modèle, qui sont : la minimisation des distances à parcourir par les cavaliers gerbeurs, la minimisation du nombre de remaniements, et le regroupement des conteneurs qui appartiennent à une même catégorie. Une démonstration de la complexité du problème de stockage de conteneurs est aussi donnée dans ce chapitre.

Pour la résolution numérique, deux cas sont distingués : le cas où la cour de stockage est assez grande pour que tous les conteneurs puissent être stockés sans provoquer aucun remaniement, et le cas où les remaniements sont inévitables. Un algorithme de branch-and-cut, qui est une méthode de résolution exacte est proposé, de même qu'un algorithme de colonie d'abeilles, qui est une méta-heuristique. Des simulations numériques ont été faites pour valider ces deux algorithmes, en les comparant au logiciel d'optimisation "ILOG CPLEX".

Le cas du stockage dynamique est traité dans le chapitre suivant.

Chapitre 5

Approches proposées pour la résolution du cas dynamique

5.1 Introduction

Dans ce chapitre, nous traitons le cas dynamique du problème de stockage de conteneurs dans un terminal portuaire. La majeure différence entre ce cas et le cas statique est le fait que, dans le cas dynamique on considère que tous les conteneurs ne sont pas encore arrivés dans le terminal au début des opérations de stockage, même si on connaît les dates de départ et d'arrivée des conteneurs.

Le reste du chapitre est organisé comme suit : une description du contexte étudié est donnée dans la section 5.2, une modélisation mathématique est proposée dans la section 5.3, une description du processus général de résolution que nous avons adopté est donnée dans la section 5.4, un algorithme de colonie de fourmis est proposé dans la section 5.5, un algorithme génétique est présenté dans la section 5.6, une hybridation entre l'algorithme génétique et l'algorithme de colonie de fourmis est décrite dans la section 5.7, un algorithme de recuit simulé est proposé dans la section 5.8, une hybridation entre le recuit simulé et l'algorithme de colonie de fourmis est présentée dans la section 5.9, une hybridation entre l'algorithme génétique et le recuit simulé est proposée dans la section 5.10, une comparaison entre les différents algorithmes est faite dans la section 5.11, une conclusion est donnée dans la section 5.12.

5.2 Description du contexte

Pour traiter le cas dynamique du stockage de conteneurs, nous utilisons la méthode de "*planification par horizon*". Cette méthode consiste à diviser une longue période du temps en plusieurs parties, chaque partie représente une période de travail. Par exemple, un jour peut être divisé en quatre périodes de 6 heures, comme le montre la figure suivante.

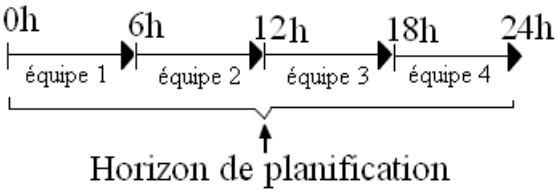


FIGURE 32 – Exemple

Au début de la période de travail de chaque équipe, les conteneurs qui partent sont d'abord enlevés avant le stockage des conteneurs qui arrivent. Cependant, si l'extraction d'un conteneur nécessite un remaniement, les conteneurs déplacés seront remis dans la pile d'origine.

Nous considérons un terminal multi-modal, qui reçoit et expédie des conteneurs par différents modes de transport : maritime, routier, et ferroviaire. Les conteneurs qui arrivent par navire, et qui partent par train ou bien par camion, sont appelés *conteneurs entrants* (c'est-à-dire des conteneurs qui sont importés). Par contre, les conteneurs qui arrivent par camion ou par train, et qui partent par bateau sont appelés *conteneurs sortants* (c'est-à-dire des conteneurs qui vont être exportés). Il existe aussi le cas des conteneurs qui transitent entre différents bateaux, on les appelle des *conteneurs en transbordement*. Chacun de ces conteneurs séjourne temporairement dans la cour de stockage du port. Une façon de faire consiste à diviser la cour de stockage en plusieurs zones, qui sont chacune réservée à un type de conteneur précis. Cependant, tous les travaux de recherche qui ont exploité cette méthode ont souligné qu'elle comporte un risque de gaspillage d'espace et qu'elle ne favorise pas une bonne exploitation de la cour de stockage. Ainsi pour éviter ce désagrément, nous proposons une stratégie de stockage qui autorise le stockage de différents conteneurs (entrant, sortant, ou en transbordement) dans chaque emplacement. Cette stratégie a pour objectif de déterminer un emplacement de stockage précis pour chaque conteneur qui arrive au terminal, d'une manière efficace qui minimise le nombre total de remaniements et les distances à parcourir entre les emplacements de stockage et les portes de sortie des conteneurs. Nous considérons que la cour de stockage dispose de plusieurs entrées (ou sorties), chacune d'elle étant connectée à une voie de transport. Pour accélérer les opérations de chargement de trains et de navires, de même que la livraison des conteneurs aux camions, nous privilégions le regroupement des conteneurs qui ont les mêmes propriétés (dimension, mode de transport, et heure de départ) dans des piles adjacentes. Au meilleur de nos connaissances, il n'existe pas de travail précédent qui traite le cas dynamique du problème de stockage de conteneurs en affectant chaque conteneur à un emplacement de stockage précis.

Comme pour le cas statique du stockage de conteneurs, nous considérons un terminal à conteneurs qui utilise des cavaliers gerbeurs comme équipement de manutention et de transfert. Les hypothèses qui sont considérées sont les suivantes :
1) Pour tenir compte des différences entre les tailles des conteneurs, et éviter les problèmes de déséquilibre, nous stockons dans chaque pile uniquement des conteneurs qui ont des dimensions similaires.
2) Pour éviter l'éparpillement des conteneurs dans la cour de stockage, et aussi pour ac-

célérer les opérations d'extradition de conteneurs vers leurs moyens de transport, nous divisons ces derniers en plusieurs catégories. Chaque groupe est constitué de conteneurs qui ont les mêmes dimensions, les mêmes dates de départ, et qui seront chargés sur le même train ou navire, ou bien qui appartiennent à un même client dans le cas des conteneurs qui partent par camions. Le but de ce procédé est de regrouper autant que possible les conteneurs de chaque catégorie.
3) Les dates d'arrivée et les ordres de déchargement des conteneurs sont connus avant le début de l'horizon de planification. Ainsi, par commodité, on suppose que les conteneurs sont numérotés suivant l'ordre croissant de leurs périodes d'arrivée, et que des conteneurs qui sont déchargés d'un même moyen de transport sont numérotés suivant leur ordre de déchargement.
4) On suppose que les piles sont numérotées de sorte que deux piles adjacentes qui sont dans une même travée aient des numéros successifs, et que si deux travées sont adjacentes alors le numéro de la dernière pile de l'une succède au numéro de la première pile de l'autre (voir Figure 23).
5) On considère qu'une journée est composée de plusieurs périodes de stockage.

5.3 Modélisation mathématique

Pour la résolution du cas dynamique du problème de stockage de conteneurs, nous proposons le modèle mathématique suivant, dans lequel sont utilisés les éléments contenus dans le Tableau 5.1.

Tableau 5.1: Paramètres

	Notation	Description
Indice	k	Conteneur
	p	Pile
	i	Emplacement dans une pile
	h	Partie (période) de l'horizon de planification
Données concernant les conteneurs	A_0	L'ensemble des conteneurs qui sont déjà dans la cour de stockage au début de l'horizon de planification
	A_h	L'ensemble des conteneurs qui arrivent au port durant la période h
	D_h	L'ensemble des conteneurs qui quittent le port durant la période h
	R_k	Dimension du conteneur k (20 pieds, ou 40 pieds, ou 45 pieds)
	T_k	Période de départ du conteneur k
	V_k	Catégorie à laquelle appartient le conteneur k
	L_k	Période d'arrivée du conteneur k
Données concernant les piles	N_p	Nombre de piles
	c_p	Nombre d'emplacements libres dans p
	r_p	Dimension de la pile p (20 pieds, ou 40 pieds, ou 45 pieds)

	P_0^k	La pile qui contient le conteneur $k \in A_0$
	I_0^k	La position du conteneur $k \in A_0$, dans la pile où il est stocké
Données générales	d_p^k	Distance entre la pile p et la porte de sortie du conteneur k
	M	Un grand nombre entier
	H	Nombre de parties (périodes) dans l'horizon de planification
	c_{max}	Le nombre maximal de conteneurs qui peuvent séjourner simultanément dans une pile

Nous utilisons ces quatre variables de décision suivantes dans le modèle mathématique que nous proposons.

$$x_{p,i}^{k,h} = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à l'emplacement } i \text{ de la pile } p \\ & \text{durant la période } h \\ 0 & \text{Sinon} \end{cases}$$

$$y_p^k = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à la pile } p, \\ & \text{et que cela cause un remaniement} \\ 0 & \text{Sinon} \end{cases}$$

$z_k^{k'} \in \mathbb{N}$: la distance (en nombre de piles) entre les piles où sont stockés deux conteneurs k et k' qui appartiennent à une même catégorie. Si k et k' n'appartiennent pas à la même catégorie, ou bien s'ils sont stockés dans une même pile, alors $z_k^{k'} = 0$.

$c_p^h \in \mathbb{N}$: le nombre d'emplacements inoccupés qui sont dans la pile p à la fin de la période h.

À partir de ces données, nous construisons le modèle mathématique suivant.

$$Minimiser \sum_{h=1}^{H} \sum_{k \in A_h} \sum_{p=1}^{N_p} \sum_{i=0}^{c_{max}} d_p^k x_{p,i}^{k,h} + \sum_{k=1}^{N} \sum_{p=1}^{N_p} M y_p^k + \sum_{k=1}^{N} \sum_{k'=k+1}^{N} z_k^{k'} \tag{5.1}$$

La fonction objectif (5.1) minimise simultanément, pour toutes les périodes qui composent l'horizon de planification, le nombre de remaniements et la distance totale entre les emplacements de stockage et les sorties. En plus de cela, elle minimise les distances entre les conteneurs qui appartiennent à une même catégorie.

$$x_{P_0^k, I_0^k}^{k,0} = 1, \quad \forall\, k \in A_0 \tag{5.2}$$

La contrainte (5.2) précise les emplacements de stockage qui sont déjà occupés au début de l'horizon de planification.

$$\sum_{p=0}^{N_p}\sum_{i=0}^{c_{max}} x_{p,i}^{k,L_k} = 1, \quad \forall\, k \in \cup_{h=0}^{H} A_h \tag{5.3}$$

La contrainte (5.3) assure que chaque conteneur est affecté à un seul emplacement de stockage à son arrivée.

$$\sum_{p=1}^{N_p}\sum_{i=1}^{c_{max}} x_{p,i}^{k,h} = 0, \quad \forall\, k \in \cup_{h=0}^{H} A_h, \text{tel que } h < L_k \text{ ou } h \geq T_k; \tag{5.4}$$

La contrainte (5.4) garantit que les emplacements de stockage sont réservés uniquement durant les temps de séjour des conteneurs. En d'autres termes, aucun emplacement n'est réservé pour un conteneur donné, durant les périodes qui précédent son arrivée et celles qui succèdent son départ.

$$\begin{aligned}\sum_{i=1}^{c_{max}} x_{p,i}^{k,L_k} \geq \sum_{i=1}^{c_{max}} x_{p,i}^{k,h}, \quad \forall\, p = 1,...,N_p;\ k \in \cup_{h=0}^{H} A_h \\ \forall\, h = L_k + 1, ..., \min\{D_k, H\}\end{aligned} \tag{5.5}$$

La contrainte (5.5) assure que chaque conteneur reste dans la pile où il est affecté à son arrivée, durant toute la durée de son séjour. Ce qui signifie que même après un remaniement, les conteneurs déplacés sont remis dans leurs piles d'origine.

$$\begin{aligned}\sum_{i=1}^{c_{max}} i x_{p,i}^{k,h} - \sum_{i=1}^{c_{max}} i x_{p,i}^{k,h'} \geq M\left(\sum_{i=1}^{c_{max}} x_{p,i}^{k,h} - 1\right) \\ \forall\, p = 1,...,N_p;\ k \in \cup_{h=0}^{H} A_h;\ h = L_k, ..., \min\{T_k, H\}; \\ h' = h+1, ..., \min\{T_k, H\}\end{aligned} \tag{5.6}$$

La contrainte (5.6) assure qu'un conteneur ne peut pas avoir une position de stockage plus élevée que celle qui lui était assignée à son arrivée. En d'autres termes, après un remaniement, les conteneurs qui étaient au-dessus du conteneur retiré seront descendus vers le bas.

$$\sum_{k \in \cup_{h=0}^{H} A_h} x_{p,i}^{k,h} \leq 1, \quad \forall\, p = 1,...,N_p;\ \ h = 0,...,H;\ i = 1,...,c_{max} \tag{5.7}$$

La contrainte (5.7) garantit qu'à chaque période de l'horizon de planification, au plus un conteneur est affecté à chaque emplacement de stockage.

$$\sum_{k \in A_h}\sum_{i=1}^{c_{max}} x_{p,i}^{k,L_k} \leq c_p^{h-1} + \sum_{k \in D_h}\sum_{i=1}^{c_{max}} x_{p,i}^{k,L_k}, \quad \forall\, h = 1,...,H;\ p = 1,...,N_p \tag{5.8}$$

La contrainte (5.8) assure que les capacités des piles ne sont pas dépassées. Ce qui signifie, qu'à chaque période, le nombre de nouveaux conteneurs affectés à une pile est inférieur ou égal à la somme d'emplacements inoccupés dans cette dernière, après l'extraction des

conteneurs qui doivent partir. Rappelons qu'à chaque période de l'horizon de planification, les conteneurs qui doivent quitter le terminal sont retirés avant le stockage des nouveaux conteneurs.

$$c_p^h = c_p^{h-1} - \sum_{k \in A_h} \sum_{i=1}^{c_{max}} x_{p,i}^{k,h} + \sum_{k \in D_h} \sum_{i=1}^{c_{max}} x_{p,i}^{k,L_k} \qquad \forall\, p = 1, ..., N_p : \; h = 1, ..., H \tag{5.9}$$

La contrainte (5.9) met à jour la capacité de chaque pile, à la fin de chaque période de l'horizon de planification.

$$\sum_{h=0}^{H} \sum_{i=0}^{c_{max}} x_{p,i}^{k,h} = 0, \quad \forall\, k \in \cup_{h=0}^{H} A_h;\; p = 1, ..., N_p, \text{ tel que } R_k \neq r_p \tag{5.10}$$

La contrainte (5.10) assure que les conteneurs qui sont affectés à une même pile ont les mêmes mesures.

$$\sum_{i=0}^{c_{max}} i x_{p,i}^{k,L_k} \geq c_{max} - c_p^{h-1} + 1 - \sum_{k' \in D_h} \sum_{i=0}^{c_{max}} x_{p,i}^{k',L_{k'}} + M(\sum_{i=0}^{c_{max}} x_{p,i}^{k,L_k} - 1) \qquad \forall\, h = 1, ..., H;\; p = 1, ..., N_p;\; k \in A_h \tag{5.11}$$

La contrainte (5.11) précise qu'à chaque période, un nouveau conteneur qui est affecté à une pile ne peut pas avoir une position de stockage plus basse que celle du conteneur qui est déjà au sommet de la pile. Autrement dit, si deux conteneurs qui ont différentes périodes d'arrivées se croisent dans une pile, alors le dernier sera forcément au-dessus de l'autre.

$$i x_{p,i}^{k,L_k} \leq c_{max} - c_p^{h-1} + 1 - \sum_{k' \in D_h} \sum_{i'=0}^{c_{max}} x_{p,i'}^{k',L_{k'}} + \sum_{k' \in A_h} (i-1) x_{p,i-1}^{k',L_{k'}} \qquad \forall\, h = 1, ..., H;\; k \in A_h;\; p = 1, ..., N_p;\; i = 2, ..., c_{max} \tag{5.12}$$

La contrainte (5.12) permet de satisfaire les lois de la pesanteur. Ce qui veut dire que le remplissage de chaque pile se fait de bas en haut sans sauter aucun emplacement intermédiaire.

$$\sum_{i=0}^{c_{max}} i x_{p,i}^{k,L_k} - \sum_{i=0}^{c_{max}} i x_{p,i}^{k',L_{k'}} \leq M(1 - \sum_{i=1}^{c_{max}} x_{p,i}^{k',L_{k'}}) \qquad \forall\, h = 1, ..., H;\; p = 1, ..., N_p;\; k \in A_h;\; k' \in A_h \text{ et } k' > k \tag{5.13}$$

La contrainte (5.13) assure que si des conteneurs qui arrivent durant une même période sont affectés à une même pile, alors ils seront stockés suivant la loi du *premier arrivé - premier stocké*. Cela permet d'éviter les congestions sur les quais et aux alentours des

rails, car les conteneurs seront ramassés suivant leur ordre de déchargement. En plus de cela, dans le cas des conteneurs qui sont apportés au terminal par des camions, cette contrainte permet d'éviter des favoritismes entre clients. Rappelons que les conteneurs sont numérotés suivant le même ordre que leurs arrivées et leurs déchargements.

$$z_k^{k'} = \left| \sum_{p=1}^{N_p} \sum_{i=0}^{c_{max}} p x_{p,i}^{k,L_k} - \sum_{p=1}^{N_p} \sum_{i=0}^{c_{max}} p x_{p,i}^{k',L_{k'}} \right|, \quad \forall\ k \in \cup_{h=0}^{H} A_h;\ k' \in \cup_{h=L_k}^{T_k-1} A_h \text{ et } V_k = V_{k'} \tag{5.14}$$

La contrainte (5.14) calcule la distance qui sépare deux conteneurs qui appartiennent à une même catégorie. Rappelons que notre objectif est de regrouper autant que possible, dans une même pile ou bien dans des piles adjacentes, les conteneurs qui appartiennent à une même catégorie (dimension, destination, moyen de transport, et période de départ similaires).

$$T_k x_{p,i}^{k,L_k} - T_{k'} \sum_{h=L_{k'}}^{L_k} x_{p,i-1}^{k',h} \leq M \left(y_p^k + 1 - \sum_{h=L_{k'}}^{L_k} x_{p,i-1}^{k',h} \right) \quad \forall\ p = 1, ..., N_p;\ i = 2, ..., c_{max};\ k' \in \cup_{h=0}^{H} A_h;\ k \in \cup_{h=L_{k'}}^{T_{k'}-1} A_h \tag{5.15}$$

La contrainte (5.15) force que y_p^k soit égale à 1, si le conteneur k est affecté à la pile p, sachant que sa date de départ est supérieure à celle d'un conteneur qui est déjà à l'intérieur de p. Cette contrainte permet de compter le nombre de remaniements.

$$\sum_{k \in A_h} T_k x_{p,i}^{k,L_k} - \sum_{k \in A_h} T_k x_{p,i+1}^{k,L_k} \geq M \left(\sum_{k \in A_h} x_{p,i}^{k,L_k} - 1 \right) \quad \forall\ p = 1, ..., N_p;\ i = 1, ..., c_{max} - 1;\ h = 1, ..., H \tag{5.16}$$

La contrainte (5.16) assure que si plusieurs conteneurs qui arrivent durant une même période sont affectés à une même pile, alors ils seront stockés suivant l'ordre décroissant de leurs dates de départ. Cela permet de minimiser le nombre de remaniements, voire même les éliminer complètement dans le cas où la cour de stockage est assez vaste.

$$\sum_{h=L_k}^{T_k-1} x_{p,1}^{k,h} + \sum_{h=L_{k'}}^{T_k-1} x_{p,2}^{k',h} - x_{p,1}^{k',T_k} < 2 \quad \forall\ p = 1, ..., N_p;\ k \in \cup_{h=0}^{H} A_h;\ k' \in \cup_{h=L_k}^{T_k-1} A_h;\ T_k < T_{k'} \text{ et } T_k \leq H \tag{5.17}$$

La contrainte (5.17) assure que si un conteneur qui est au fond d'une pile a une date de départ inférieure à celle du conteneur qui est juste au-dessus de lui, alors, après le départ du premier conteneur, le deuxième sera placé au fond de la pile. La figure (33) est une illustration de cette contrainte.

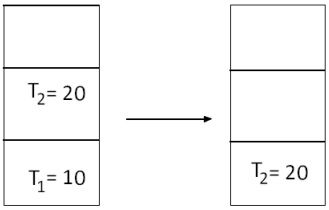


FIGURE 33 – Déplacement d'un conteneur qui n'était ni au sommet, ni au fond, d'une pile

$$\begin{aligned} \sum_{h'=L_k}^{h} x_{p,i}^{k,h'} + \sum_{h'=L_{k'}}^{h} x_{p,i+1}^{k',h'} + x_{p,i-1}^{k,h+1} - x_{p,i}^{k',h+1} < 3 \\ \forall\ k \in \cup_{h=0}^{H} A_h;\ k' \in \cup_{h=L_k}^{T_k-1} A_h;\ h = L_k, ..., \min\{T_{k'} - 2, H\} \\ \forall\ p = 1, ..., N_p;\ i = 2, ..., c_{max} - 1 \end{aligned} \tag{5.18}$$

La contrainte (5.18) assure que si on change (déplacer vers le bas) l'emplacement de stockage d'un conteneur, alors on change (déplacer vers le bas) également celui du conteneur qui est juste au-dessus. La figure (34) est une illustration de cette contrainte.

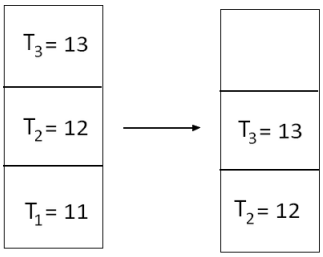


FIGURE 34 – Déplacement d'un conteneur vers le bas

$$\begin{aligned} \sum_{h=L_k}^{T_{k'}-1} x_{p,1}^{k,h} + \sum_{h=L_{k'}}^{T_{k'}-1} x_{p,2}^{k',h} + \sum_{h=L_{k''}}^{T_{k'}-1} x_{p,3}^{k'',h} - x_{p,2}^{k'',T_{k'}} < 3 \\ \forall\ p = 1, ..., N_p;\ k \in \cup_{h=0}^{H} A_h; k' \in \cup_{h=L_k}^{T_k-1} A_h;\ \ k'' \in \cup_{h=L_{k'}}^{T_{k'}-1} A_h \\ T_{k'} < T_k;\ T_{k'} < T_{k''};\ \text{et}\ T_{k'} \leq H \end{aligned} \tag{5.19}$$

La contrainte (5.19) assure que si le conteneur qui est stocké dans le deuxième emplacement d'une pile a une date de départ inférieure à celles des conteneurs qui sont au fond et au sommet, alors, après son départ, le conteneur qui était au sommet sera mis dans le deuxième emplacement. La figure (35) est une illustration de cette contrainte.

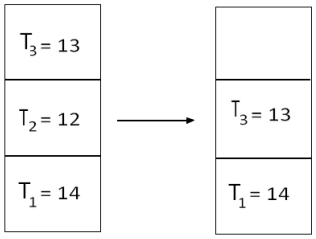


FIGURE 35 – Repositionnement d'un conteneur

$$\begin{gathered} \sum_{h=L_k}^{T_k-1} x_{p,1}^{k,h} + \sum_{h=L_{k'}}^{T_k-1} x_{p,2}^{k',h} + \sum_{h=L_{k''}}^{T_k-1} x_{p,3}^{k'',h} - x_{p,1}^{k'',T_k} < 3 \\ \forall\ p = 1, ..., N_p;\ k \in \cup_{h=0}^{H} A_h;\ k' \in \cup_{h=L_k}^{T_k-1} A_h;\ k'' \in \cup_{h=L_{k'}}^{T_{k'}-1} A_h \\ T_{k'} = T_k;\ T_k < T_{k''};\ \text{et}\ T_k \leq H \end{gathered} \tag{5.20}$$

La contrainte (5.20) garantit que si le conteneur qui est fond d'une pile et celui qui est au milieu ont une même date de départ qui est inférieure à celle du conteneur qui est au sommet, alors ce dernier sera placé au fond de la pile quand les deux autres seront enlevés. La figure (36) est une illustration de cette contrainte.

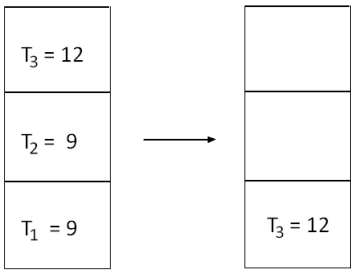


FIGURE 36 – Déplacement d'un conteneur qui était au sommet d'une pile

$$\begin{gathered} x_{p,i}^{k,h} \in \{0,1\},\ y_p^k \in \{0,1\}, c_p^h \in \mathbb{N},\ \text{et}\ z_k^{k'} \in \mathbb{N} \\ \forall\ p = 1, ..., N_p,\ i = 1, ..., c_p,\ k = 1, ..., N,\ h = 1, ..., H \end{gathered} \tag{5.21}$$

La contrainte (5.21) précise la nature de chaque variable de décision.

5.4 Processus général de résolution

Dans les algorithmes que nous proposons pour la résolution du cas dynamique du problème de stockage de conteneurs, nous considérons que le plan de stockage des conteneurs qui arrivent durant chaque période de l'horizon de planification est déterminé en tenant compte de l'état du terminal à la fin de la période précédente. Cela sous-entend que la

détermination des emplacements de stockage de tous les conteneurs qui arrivent durant un horizon de planification se fait séquentiellement en traitant successivement les conteneurs qui arrivent durant chaque période. Ainsi, chacun de nos algorithmes contiendra au moins autant d'itérations qu'il y a de périodes dans l'horizon de planification. De façon générale, un algorithme prend en entrée un fichier qui contient deux types d'informations capitales, à savoir :

1) Les emplacements de stockage qui sont déjà occupés ainsi que les informations (taille, date de départ, catégorie) concernant les conteneurs qui les occupent.
2) Les informations (taille, date d'arrivée, date de départ, catégorie) sur les conteneurs qui arrivent durant l'horizon de planification.

Chaque algorithme renvoie en sortie un fichier qui contient la meilleure solution qu'il a trouvé, c'est-à-dire un fichier qui renseigne les emplacements de stockage qui sont affectés aux conteneurs arrivés durant l'horizon de planification. Ce fichier précise également la valeur de la fonction objectif, la distance totale, le nombre total de remaniements, et la durée d'exécution.

La figure 37 est une illustration du processus de résolution.

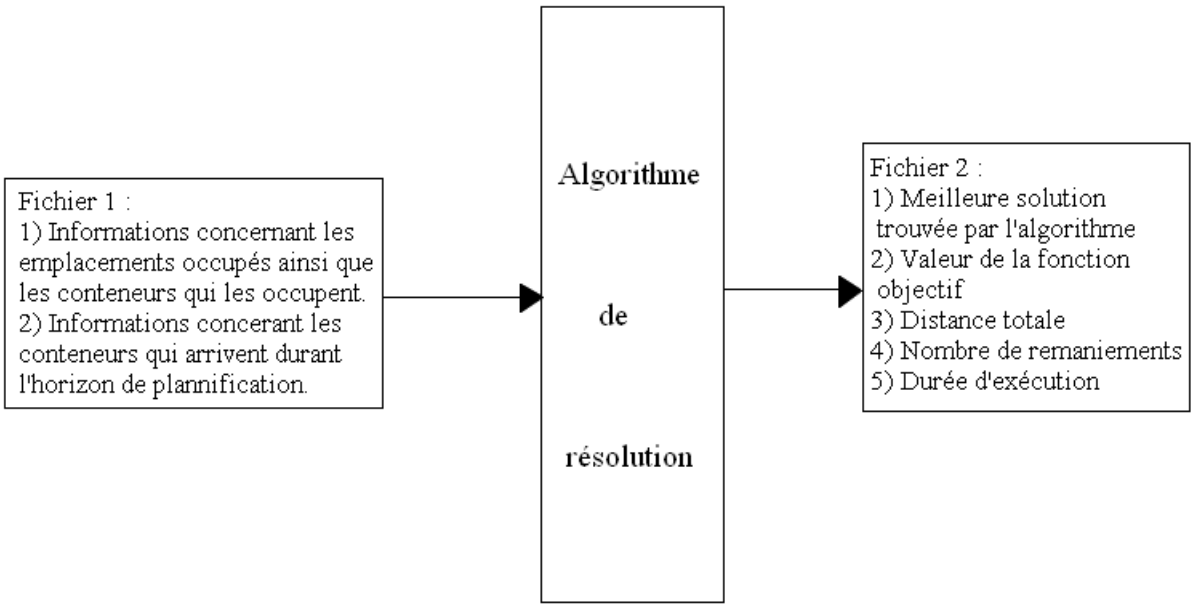


FIGURE 37 – Schéma général de résolution

5.5 Algorithme de colonie de fourmis

Nous avons utilisé les quatre variantes de l'algorithme de colonie de fourmis pour la résolution du problème de stockage de conteneurs, à savoir : l'*Ant System* (AS), le *rank-based* (AS_{rank}), le *MAX-MIN Ant System* (MMAS), et le *Ant Colony System* (ACS). La différence majeure entre ces variantes se trouve au niveau de la méthode d'augmentation de la phéromone.

Comme pour l'algorithme de colonie d'abeilles qui est présenté dans la section 4.6.3, nous représentons une solution sous la forme d'un tableau qui a deux lignes et dont le nombre de colonnes est égal au nombre de conteneurs qui arrivent durant la période courante. Notons que la solution obtenue pour tout l'horizon de planification est composée des solutions des différentes périodes, par conséquent son nombre de colonnes est égal au nombre total de conteneurs qui arrivent durant l'horizon de planification. Par exemple, supposons qu'on ait 15 conteneurs qui doivent arriver durant un horizon de planification composé de 3 périodes réparties comme suit : 5 conteneurs pour la première période, 7 pour la deuxième période, et 3 pour la troisième période. Une solution pour tout l'horizon de planification est composée de trois sous-solutions différenciées par des couleurs qui représentent chacune une période (voir l'exemple de la figure 38). La partie en vert est pour la période 1, celle en rose est pour la période 2, alors que celle en bleu est pour la période 3. Les conteneurs sont marqués dans la première ligne alors que les piles sont notées dans la deuxième.

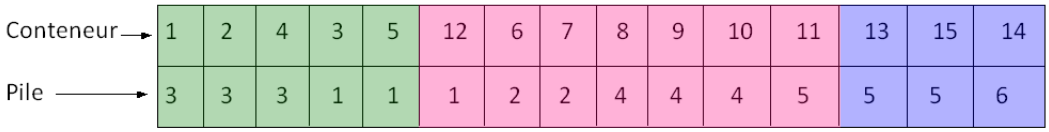


FIGURE 38 – Représentation d'une solution

Si plusieurs conteneurs sont affectés à une même pile, alors ils seront stockés suivant l'ordre croissant de leurs numéros de colonne. Cela permet de tenir en compte, dans le codage, le fait que dans chaque pile les conteneurs sont stockés suivant l'ordre croissant de leurs dates d'arrivée dans le terminal et que des conteneurs qui arrivent au cours d'une même période sont stockés suivant l'ordre décroissant de leurs dates de départ afin d'éviter de causer des remaniements. Dans l'exemple précédent, les conteneurs 1, 2, et 4 seront disposés dans la pile 3 comme suit.

FIGURE 39 – Disposition des conteneurs

Nous utilisons cette représentation de solution dans tous les algorithmes que nous proposons dans ce chapitre.

La figure 40 est une représentation de l'algorithme de colonie de fourmis.

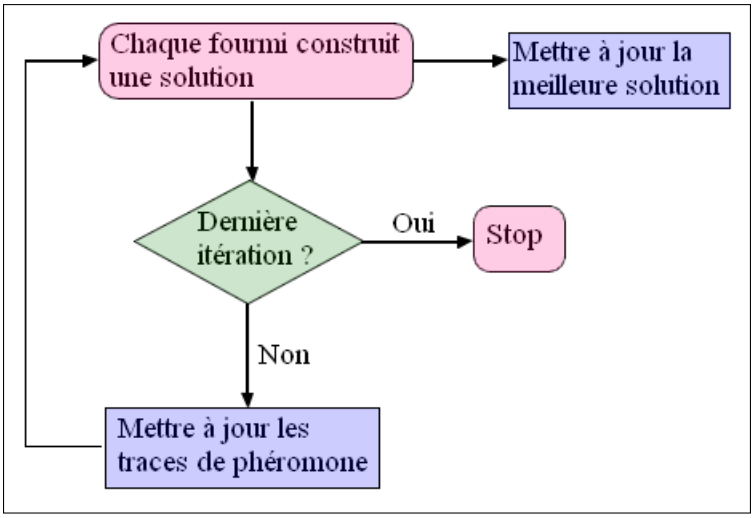


FIGURE 40 – Algorithme de colonie de fourmis

L'algorithme de colonie de fourmis est composé de plusieurs itérations dont chacune contient principalement deux étapes qui sont : la construction des solutions par les fourmis et la mise à jour des traces de phéromones.

5.5.1 Méthode de construction d'une solution par une fourmi

En observant la représentation de solution que nous avons adopté dans nos algorithmes de colonie de fourmis (voir Figure 38), on peut constater que chaque solution est une succession de couples (conteneur, pile) qui forment chacun une colonne de la solution. Une solution valide doit inclure tous les conteneurs, en plus de cela les conteneurs des couples doivent être deux à deux distincts. Donc, dans une solution valide on a autant de couples que de conteneurs à stocker. Ainsi, avant de commencer l'algorithme de colonie de fourmis, il faut d'abord rechercher tous les couples (conteneur "k", pile "p") qui sont compatibles, en d'autres termes, qui vérifient ces deux conditions suivantes :

- la pile n'est pas pleine ($c_p > 0$),
- le conteneur et la pile ont les mêmes mesures ($r_p = R_k$).

Notons E l'ensemble des couples qui sont compatibles. Chaque élément (k, p) de E a une trace de phéromone $\tau_{(k,p)}$ qui est initialisée à 1. Dans la procédure de construction d'une solution, nous utilisons la formule suivante pour calculer la probabilité de sélection de chaque couple.

$$P_{(k,p)} = \frac{(\tau_{(k,p)})^{\alpha} \times (\frac{1}{d_p^k})^{\beta}}{\sum_{(k,p)\in E_j} (\tau_{(k,p)})^{\alpha} \times (\frac{1}{d_p^k})^{\beta}}, \quad \forall (k,p) \in E_j \qquad (a)$$

où $E_j \subseteq E$, $0 < \alpha < 1$, et $0 < \beta < 1$
d_p^k est la distance entre la pile p et la porte de sortie du conteneur k
$\frac{1}{d_p^k}$ est la visibilité du couple (k, p).

Chaque fourmi construit séquentiellement sa solution (que l'on nomme S), en y ajoutant un par un des couples qui appartiennent à E. La procédure de construction d'une solution est la suivante.

1. Initialisation, $S = \emptyset$, $Compteur = 0$
2. Tant que ($NombreCouplesDans(S) < NombreConteneurs$
   et $Compteur < NombreEssaisMaximal$) faire
   - 2.a. Rassembler les couples qui ne causent aucun remaniement
     $E_1 = \{(k,p) \in E : t_p \geq T_k\}$
   - 2.b. Rassembler les couples qui causent des remaniements
     $E_2 = \{(k,p) \in E : t_p < T_k\}$
   - 2.c. Si ($E_1 \neq \emptyset$) alors
     - 2.c.1. Choisir aléatoirement un élément dans E_1, puis l'ajouter dans S
     - 2.c.2. Tant que ($E_1 \neq \emptyset$) faire
       - 2.c.2.a Mettre à jour E_1 et E_2
       - 2.c.2.b Calculer la probabilité de tout élément restant dans E_1
       - 2.c.2.c Ajouter dans S l'élément de E_1 qui a la plus grande probabilité
     - 2.c.3. Fin tant que
     - 2.c.4. Si ($NombreCouplesDans(S) < NombreConteneurs$ et $E_2 \neq \emptyset$)
       - 2.c.4.a. Tant que ($E_2 \neq \emptyset$) faire
         - 2.c.4.a.1. Calculer la probabilité de tout élément restant dans E_2
         - 2.c.4.a.2. Ajouter dans S l'élément de E_2 qui a la plus grande probabilité
         - 2.c.4.a.3. Mettre à jour E_2
       - 2.c.4.b. Fin Tant que
   - 2.d. Sinon
     - 2.d.1. Si ($E_2 \neq \emptyset$)
       - 2.d.1.a. Choisir aléatoirement un élément dans E_2, puis l'ajouter dans S
       - 2.d.1.b. Tant que ($E_2 \neq \emptyset$) faire
         - 2.d.1.b.1. Mettre à jour E_2
         - 2.d.1.b.2. Calculer la probabilité de tout élément restant dans E_2
         - 2.d.1.b.3. Ajouter dans S l'élément de E_2 qui a la plus grande probabilité
       - 2.d.1.c. Fin tant que
     - 2.d.2. Fin Si
   - 2.e. Fin Sinon
   - 2.f. Si ($NombreCouplesDans(S) < NombreConteneurs$) alors
     - 2.f.a. $S = \emptyset$
     - 2.f.b. $Compteur = Compteur + 1$
   - 2.g. Fin Si
3. Fin tant que
4. Si ($NombreCouplesDans(S) = NombreConteneurs$) alors
   - 4.a. Ordonner les couples de S qui ont la même pile suivant l'ordre croissant de leurs numéros de conteneur et l'ordre décroissant des dates de départ de leurs conteneurs
5. Fin Si

Dans le pseudo-code précédent, T_k est la date de départ du conteneur k. Alors que,

t_p est la date de départ du conteneur qui était au sommet de la pile p, au début des opérations de stockage de la période considérée de l'horizon de planification.

Avant de commencer à construire une solution, l'ensemble des couples E est divisé en deux sous-ensembles : E_1 et E_2. Le premier sous-ensemble contient les couples qui ne comportent aucun risque de remaniement, c'est-à-dire tout couple (k, p) qui vérifie $t_p \geq T_k$. Alors que le deuxième sous-ensemble contient les couples qui correspondent à des affectations qui causent des remaniements. Ce procédé permet de privilégier les couples qui ne riment pas avec des remaniements par rapport aux autres. Ainsi, on commence la construction de chaque solution en considérant uniquement les éléments de E_1. Si E_1 devient vide alors que tous les conteneurs ne sont pas encore affectés, alors on utilise les éléments de E_2.

Le premier couple qui est sélectionné lors de la construction d'une solution est choisi de façon aléatoire dans E_1, si $E_1 \neq \emptyset$, sinon on le choisit dans E_2. Ensuite, tous les autres éléments qui suivent sont choisis en utilisant la formule de probabilité (a). Cependant, à chaque fois qu'un élément (k, p) est ajouté dans la solution, on diminue la capacité de la pile $(c_p = c_p - 1)$. Ensuite, on met à jour l'ensemble des couples qui sont candidats, en éliminant ceux qui risquent de rendre la solution invalide. Cela veut dire qu'on supprime, dans E_1 et dans E_2, tout couple (k', p') qui satisfait l'une des quatre conditions suivantes :

(1) $c_{p'} = 0$, car la pile p' est pleine,
(2) $k = k'$, car le conteneur k est déjà affecté à un emplacement de stockage,
(3) $k < k'$ et $T_k < T_{k'}$, car les ordres d'arrivée et de départ ne sont pas compatibles,
(4) $k > k'$ et $T_k > T_{k'}$, car les ordres d'arrivée et de départ ne sont pas compatibles.

L'utilité d'éliminer les couples qui vérifient les conditions (3) ou (4) se justifie par le fait que si de tels couples sont ajoutés dans la solution, alors, à la fin de la construction de S, les couples qui ont un même numéro de pile parmi ceux qui sont sélectionnés ne pourront pas être ordonnés suivant l'ordre croissant de leurs numéros de conteneur (c'est-à-dire l'ordre croissant de leurs dates d'arrivée puisque que les conteneurs sont numérotés suivant cet ordre) et l'ordre décroissant des dates de départ de leurs conteneurs.

L'ajout d'éléments dans la solution S ne s'arrête que lorsque E_1 et E_2 deviennent vides chacun. À la fin, on teste si tous les conteneurs sont présents dans la solution, c'est-à-dire s'il y a autant de couples dans S que de conteneurs à stocker durant la période de travail courante. Si tel n'est pas le cas, alors la solution n'est pas valide, donc on la supprime, et on recommence la construction d'une autre solution si le nombre de tentatives ($Compteur$) est inférieur au nombre d'essais autorisés ($NombreEssaisMaximal$).

À chaque itération, lorsque toutes les fourmis finissent de construire chacune une solution, on évalue chaque solution S_j en utilisant la formule suivante.

$$f(S_j) = \sum_{(k,p)\in S_j} \left(d_p^k + y_p^k + \sum_{(k',p')\in S_j : k'>k} z_k^{k'} \right), \quad \forall\, j = 1, ..., NF,$$

où NF est le nombre total de fourmis, y_p^k et $z_k^{k'}$ sont définis comme suit.

$$y_p^k = \begin{cases} 1 & \text{Si le conteneur } k \text{ est affecté à la pile } p \\ & \text{et que cela cause un remaniement} \\ 0 & \text{Sinon} \end{cases}$$

$$z_k^{k'} = \begin{cases} |p - p'| & \text{Si les conteneurs } k \text{ et } k' \text{ appartiennent au même catégorie} \\ 0 & \text{Sinon} \end{cases}$$

Si on trouve une solution qui est plus performante que la meilleure solution courante, alors on fait une mise à jour de la meilleure solution.

5.5.2 Méthodes de mise à jour des traces de phéromone

La mise à jour des traces de phéromone sert à modifier la quantité de phéromone de chaque couple (conteneur, pile) au cours des itérations de l'algorithme. Elle se fait en deux étapes : l'évaporation et l'augmentation.

L'évaporation se fait de la même façon pour toutes les variantes de l'algorithme de colonie de fourmis. Elle est effectuée à la fin de chaque itération sur tous les couples (conteneur, pile) qui appartiennent à l'ensemble E, en utilisant la formule suivante.

$$\tau_{(k,p)} = (1 - \rho)\tau_{(k,p)}, \quad \forall\, (k,p) \in E$$

Quant à l'ajout, il est fait différemment selon la version de l'algorithme de colonie de fourmis considérée.

Méthode d'ajout de phéromone pour la variante "Ant system"

Avec cette variante de l'algorithme de colonie de fourmis, à la fin de chaque itération, on augmente les quantités de phéromone des couples qui sont sélectionnés par les fourmis lors de la construction des solutions. Le pseudo-code suivant décrit cette procédure.

1. $j = 1$
2. Tant que $(j \leq NF)$ faire
   2.a. $\forall\, (k,p) \in S_j,\ \tau_{(k,p)} = \tau_{(k,p)} + \frac{1}{f(S_j)}$
   2.b. $j = j + 1$
3. Fin tant que

Avec cette méthode, plus le nombre de fourmis qui choisissent un couple est grand, plus la quantité de phéromone de ce couple est élevée. En plus de cela, une quantité additionnelle de phéromone est ajoutée sur les couples qui appartiennent à la meilleure solution (S_{Best}) de toutes les itérations déjà effectuées, en utilisant la formule suivante.

$$\tau_{(k,p)} = \tau_{(k,p)} + \frac{e}{f(S_{Best})} \quad \forall\, (k,p) \in S_{Best}$$

où e est un nombre réel positif.

Méthode d'ajout de phéromone pour la variante "Rank-based"

Avec cette méthode, on trie les solutions suivant l'ordre croissant de leurs performances, ensuite on sélectionne les meilleures. Le nombre de solutions sélectionnées est un paramètre fixé à l'avance, notons le w. Seules les traces de phéromone des couples (conteneur, pile) qui appartiennent aux solutions sélectionnées seront augmentées, suivant le pseudo-code suivant.

1. $j = 1$
2. Tant que $(j \leq w)$ faire
   2.a. $\forall\ (k,p) \in S_j,\ \tau_{(k,p)} = \tau_{(k,p)} + \frac{w+1-j}{f(S_j)}$
   2.b. $j = j + 1$
3. Fin tant que

Après cela, on ajoute une quantité de phéromone additionnelle sur les couples qui appartiennent à la meilleure solution S_{Best} de toutes les itérations déjà effectuées, en utilisant la formule suivante.

$$\tau_{(k,p)} = \tau_{(k,p)} + \frac{w}{f(S_{Best})} \quad \forall\ (k,p) \in S_{Best}$$

Méthode d'ajout de phéromone pour la variante "Ant Colony system"

Avec cette méthode, on ajoute de la phéromone uniquement sur la meilleure solution S_{BC} de l'itération courante, en utilisant la formule suivante.

$$\tau_{(k,p)} = \tau_{(k,p)} + \frac{1}{f(S_{BC})} \quad \forall\ (k,p) \in S_{BC}$$

Méthode d'ajout de phéromone pour la variante "MAX-MIN Ant System"

Avec cette méthode, on impose que les traces de phéromone soient comprises entre deux valeurs seuils (τ_{Min} et τ_{Max}). Ainsi, juste après la création de l'ensemble des couples (conteneur, pile) qui sont compatibles, on initialise la quantité de phéromone de chaque paire à τ_{Max}. Cependant, lors de la mise à jour des quantités de phéromone, on vérifie continuellement si les seuils ne sont pas dépassés, et on fait des ajustements si nécessaire. Cela signifie que lors de l'évaporation, on vérifie pour chaque couple (k,p) si sa nouvelle quantité de phéromone est inférieure à τ_{Min}, si tel est le cas, on la rend égale à τ_{Min}. De façon similaire, lorsqu'on augmente la quantité de phéromone d'un couple, on vérifie si le résultat excède τ_{Max}, si tel est le cas on la rend égale à τ_{Max}. Avec cette variante d'algorithme de colonie de fourmis, comme pour la précédente, on augmente uniquement les quantités des traces de phéromone des couples qui appartiennent à la meilleure solution trouvée dans l'itération courante, en utilisant la formule suivante.

$$\tau_{(k,p)} = \tau_{(k,p)} + \frac{1}{f(S_{BC})} \quad \forall\ (k,p) \in S_{BC}$$

5.5.3 Résultats numériques

Dans cette section, nous comparons les résultats fournis par les quatre variantes de l'algorithme de colonie de fourmis. Pour ce faire, nous commençons d'abord par chercher les bonnes valeurs des paramètres utilisés dans ces algorithmes. Chaque paramètre est traité individuellement, en fixant les autres paramètres, et en le faisant varier. Nous avons testé plusieurs instances de tailles moyennes, et nous avons retenu la meilleure valeur pour chaque paramètre. Les résultats sont notés dans le tableau 5.3.

Tableau 5.3 – Valeurs des paramètres de l'algorithme de colonie de fourmis

Paramètre	Valeur
Nombre d'itérations	$NT_{Max} = 40$
Nombre de fourmis	$NA = 17$
Puissance de la phéromone	$\alpha = 0.3$
Puissance de la visibilité	$\beta = 0.9$
Taux d'évaporation de la phéromone	$\rho = 0.2$
Taux minimal de phéromone	$\tau_{Min} = 1$
Taux maximal de phéromone	$\tau_{Max} = 10$
Nombre de solutions sélectionnées	$w = 8$
Bonus de phéromone	$e = 8$

Pour fixer le nombre d'itérations, nous avons considéré les nombres entiers compris entre 20 et 100, et nous avons choisi celui à partir duquel la valeur de la fonction objectif ne diminue plus. De façon similaire, la recherche du nombre adéquat de fourmis a été fait en considérant les nombres entiers compris entre 10 et 100. Quant à la puissance de la phéromone, à la puissance de la visibilité, et au taux d'évaporation de la phéromone, ils ont été fixés en considérant les nombres réels compris entre 0 et 1, avec un pas de 0,1. Le taux minimal ainsi que le taux maximal de phéromone ont été traités en considérant respectivement les nombres entiers compris entre 1 et 5 et ceux qui sont compris entre 5 et 10.

Après avoir fixé les valeurs des paramètres, nous comparons les quatre variantes de l'algorithme de colonie de fourmis. Pour ce faire, nous les exécutons sur de petites instances. Ensuite, nous calculons leurs écarts relatifs par rapport aux résultats optimaux trouvés par le logiciel "ILOG CPLEX", en utilisant la formule suivante

$$gap = \frac{Obj_{Algorithme} - Obj_{CPLEX}}{Obj_{CPLEX}},$$

où $Obj_{Algorithme}$ est la valeur de la solution trouvée par l'algorithme considéré, et Obj_{CPLEX} est la valeur de la solution optimale trouvée par le logiciel "ILOG CPLEX".
Les résultats obtenus sont notés dans les tableaux 5.4, 5.5, et 5.6.

Tableau 5.4: Résultats numériques de la version ACS de l'algorithme de colonie de fourmis

Instances				CPLEX				ACS				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	27524	0 sec	17463	10	0.193893
	21	53%		23874	3.12 sec	16731	6	28114	0 sec	17089	11	0.177599
	22	43%	3	28863	17.61 sec	17724	8	31776	0 sec	17761	14	0.100925
	23	46%		23751	9.78 sec	18593	1	28856	0 sec	18800	10	0.214938
	24	70%		22529	195.91 sec	19310	3	33931	1 sec	18909	15	0.506103
	25	63%	4	21331	63.73 sec	20145	1	27435	0 sec	20406	7	0.286156
	26	36%		26962	225.02 sec	20831	3	33765	0 sec	20712	13	0.252318
	27	36%		23107	2.11 sec	21951	1	28835	0 sec	21813	7	0.24789
	28	53%		24965	341.53 sec	22700	1	31112	0 sec	22094	9	0.246225
11	20	60%	2	18129	18.45 sec	16048	2	24908	0 sec	15883	9	0.373931
	21			20823	1393.86 sec	16682	4	26817	0 sec	16783	10	0.287855
	22	36%	3	24131	155.91 sec	18040	3	25992	0 sec	17908	8	0.0771207
	23	69%		19490	19.78 sec	18360	1	30274	1 sec	18178	12	0.553309
	24	30%		32568	364.96 sec	19383	11	36419	0 sec	19305	17	0.118245
	25	48%	4	22294	17.39 sec	20103	1	30013	0 sec	19964	10	0.346237
	26	39%		23801	38.39 sec	20646	2	36597	0 sec	20570	16	0.537624
	27	45%		25252	1049.26 sec	22076	1	32917	0 sec	21873	11	0.30354
Total				404924	3920.19 sec	325261	55	515285	2 sec	325511	189	4.82391

À partir des résultats contenus dans le tableau 5.4, on constate que l'écart moyen entre les résultats fournis par la variante ACS de l'algorithme de colonie de fourmis et les résultats optimaux est égal à 0.28376.

Tableau 5.5: Résultats numériques de la version AS de l'algorithme de colonie de fourmis

Instances				CPLEX				AS				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	27524	0 sec	17463	10	0.193893
	21	53%		23874	3.12 sec	16731	6	30826	0 sec	17756	13	0.291195
	22	43%	3	28863	17.61 sec	17724	8	31869	1 sec	17849	14	0.104147
	23	46%		23751	9.78 sec	18593	1	30724	0 sec	18675	12	0.293588
	24	70%		22529	195.91 sec	19310	3	34020	0 sec	18998	15	0.510054
	25	63%	4	21331	63.73 sec	20145	1	29282	0 sec	20252	9	0.372744
	26	36%		26962	225.02 sec	20831	3	35725	0 sec	20711	15	0.325013
	27	36%		23107	2.11 sec	21951	1	28896	1 sec	21873	7	0.25053
	28	53%		24965	341.53 sec	22700	1	31145	0 sec	22125	9	0.247547
11	20	60%	2	18129	18.45 sec	16048	2	25018	0 sec	15989	9	0.379999
	21			20823	1393.86 sec	16682	4	26836	1 sec	16804	10	0.288767
	22	36%	3	24131	155.91 sec	18040	3	26088	0 sec	18040	8	0.081099
	23	69%		19490	19.78 sec	18360	1	30271	0 sec	18182	12	0.553155
	24	30%		32568	364.96 sec	19383	11	37441	0 sec	19393	18	0.149625

	25	48%	4	22294	17.39 sec	20103	1	30013	1 sec	19956	10	0.346237
	26	39%		23801	38.39 sec	20646	2	29408	0 sec	21400	8	0.235578
	27	45%		25252	1049.26 sec	22076	1	30047	0 sec	22002	8	0.189886
Total				404924	3920.19 sec	325261	55	515133	4 sec	327468	187	4.81306

Les résultats contenus dans le tableau 5.5 montrent que l'écart moyen entre les résultats fournis par la variante AS de l'algorithme de colonie de fourmis et les résultats optimaux est égal à 0.28312.

Tableau 5.6: Résultats numériques de la version MMAS de l'algorithme de colonie de fourmis

Instances				CPLEX				MMAS				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	27524	0 sec	17463	10	0.193893
	21	53%		23874	3.12 sec	16731	6	30826	0 sec	17756	13	0.291195
	22	43%	3	28863	17.61 sec	17724	8	31869	1 sec	17849	14	0.104147
	23	46%		23751	9.78 sec	18593	1	30723	0 sec	18675	12	0.293546
	24	70%		22529	195.91 sec	19310	3	34020	0 sec	18998	15	0.510054
	25	63%	4	21331	63.73 sec	20145	1	29288	0 sec	20252	9	0.373025
	26	36%		26962	225.02 sec	20831	3	35786	0 sec	20755	15	0.327275
	27	36%		23107	2.11 sec	21951	1	28893	1 sec	21869	7	0.2504
	28	53%		24965	341.53 sec	22700	1	31193	0 sec	22172	9	0.249469
11	20	60%	2	18129	18.45 sec	16048	2	25018	0 sec	15989	9	0.379999
	21			20823	1393.86 sec	16682	4	26836	1 sec	16804	10	0.288767
	22	36%	3	24131	155.91 sec	18040	3	26087	0 sec	18040	8	0.0810576
	23	69%		19490	19.78 sec	18360	1	30362	0 sec	18269	12	0.557825
	24	30%		32568	364.96 sec	19383	11	37521	0 sec	19465	18	0.152082
	25	48%	4	22294	17.39 sec	20103	1	30021	0 sec	19964	10	0.346595
	26	39%		23801	38.39 sec	20646	2	29407	0 sec	21400	8	0.235536
	27	45%		25252	1049.26 sec	22076	1	30111	0 sec	22063	8	0.19242
Total				404924	3920.19 sec	325261	55	515485	3 sec	327783	187	4.82729

À partir des résultats contenus dans le tableau 5.6, on constate que l'écart moyen entre les résultats fournis par la variante MMAS de l'algorithme de colonie de fourmis et les résultats optimaux est égal à 0.28386.

Tableau 5.7: Résultats numériques de la version AS_{rank} de l'algorithme de colonie de fourmis

Instances				CPLEX				AS_{rank}				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	27524	0 sec	17463	10	0.193893
	21	53%		23874	3.12 sec	16731	6	28100	0 sec	17055	11	0.177013
	22	43%	3	28863	17.61 sec	17724	8	31869	0 sec	17849	14	0.104147
	23	46%		23751	9.78 sec	18593	1	28741	1 sec	18705	10	0.210096

	24	70%		22529	195.91 sec	19310	3	33946	0 sec	18930	15	0.506769
	25	63%	4	21331	63.73 sec	20145	1	28375	0 sec	20333	8	0.330224
	26	36%		26962	225.02 sec	20831	3	33735	0 sec	20712	13	0.251205
	27	36%		23107	2.11 sec	21951	1	27071	1 sec	22041	5	0.17155
	28	53%		24965	341.53 sec	22700	1	31142	0 sec	22066	9	0.247426
11	20	60%	2	18129	18.45 sec	16048	2	24958	1 sec	15931	9	0.376689
	21			20823	1393.86 sec	16682	4	26819	0 sec	16785	10	0.287951
	22	36%	3	24131	155.91 sec	18040	3	25992	0 sec	17908	8	0.0771207
	23	69%		19490	19.78 sec	18360	1	30271	0 sec	18182	12	0.553155
	24	30%		32568	364.96 sec	19383	11	36332	1 sec	19280	17	0.115574
	25	48%	4	22294	17.39 sec	20103	1	29984	0 sec	19964	10	0.344936
	26	39%		23801	38.39 sec	20646	2	29072	0 sec	21045	8	0.221461
	27	45%		25252	1049.26 sec	22076	1	30053	0 sec	22002	8	0.190124
Total				404924	3920.19 sec	325261	55	503984	4 sec	326251	177	4.35933

En calculant la moyenne des écarts relatifs entre les résultats fournis par la variante AS_{rank} de l'algorithme de colonie de fourmis et les résultats optimaux, on obtient 0.25643.

La comparaison des écarts associés aux différentes variantes de l'algorithme de colonie de fourmis montre que c'est la méthode AS_{rank} qui donne en moyenne les résultats les plus proches des optimums. Une comparaison des valeurs des solutions trouvées par ces variantes est donnée dans le tableau 5.8.

Tableau 5.8: Comparaison des valeurs de la fonction objectif

Instances				ACS	AS	MMAS	AS_{rank}
Np	N	P_d	H				
10	20	33%	2	27524	27524	27524	27524
	21	53%		28114	30826	30826	28100
	22	43%	3	31776	31869	31869	31869
	23	46%		28856	30724	30723	28741
	24	70%		33931	34020	34020	33946
	25	63%	4	27435	29282	29288	28375
	26	36%		33765	35725	35786	33735
	27	36%		28835	28896	28893	27071
	28	53%		31112	31145	31193	31142
11	20	60%	2	24908	25018	25018	24958
	21			26817	26836	26836	26819
	22	36%	3	24131	25992	26087	25992
	23	69%		30274	30271	30362	30271
	24	30%		36419	37441	37521	36332
	25	48%	4	30013	30013	30021	29984
	26	39%		36597	29408	29407	29072
	27	45%		32917	30047	30111	30053

On constate, à partir du tableau 5.8, que c'est la version AS_{rank} qui donne le plus

grand nombre de meilleurs résultats.

Une comparaison des distances trouvées par les différentes versions de l'algorithme de colonie de fourmis est donnée dans le tableau 5.9.

Tableau 5.9: Comparaison des distances

Instances				ACS	AS	MMAS	AS_{rank}
Np	N	P_d	H				
10	20	33%	2	17463	17463	17463	17463
	21	53%		17089	17756	17756	17055
	22	43%	3	17761	17849	17849	17849
	23	46%		18800	18675	18675	18705
	24	70%		18909	18998	18998	18930
	25	63%	4	20406	20252	20252	20333
	26	36%		20712	20711	20755	20712
	27	36%		21813	21873	21869	22041
	28	53%		22094	22125	22172	22066
11	20	60%	2	15883	15989	15989	15931
	21			16783	16804	16804	16785
	22	36%	3	17908	18040	18040	17908
	23	69%		18178	18182	18269	18182
	24	30%		19305	19393	19465	19280
	25	48%	4	19964	19956	19964	19964
	26	39%		20570	21400	21400	21045
	27	45%		21873	22002	22063	22002

Le tableau 5.9 montre que c'est la version ACS qui trouve les plus petites distances dans la plupart des cas.

Une comparaison des nombres de remaniements trouvés par les différentes versions de l'algorithme de colonie de fourmis est donnée dans le tableau 5.10.

Tableau 5.10: Comparaison des nombres de remaniements

Instances				ACS	AS	MMAS	AS_{rank}
Np	N	P_d	H				
10	20	33%	2	10	10	10	10
	21	53%		11	13	13	11
	22	43%	3	14	14	14	14
	23	46%		10	12	12	10
	24	70%		15	15	15	15
	25	63%	4	7	9	9	8
	26	36%		13	15	15	13
	27	36%		7	7	7	5
	28	53%		9	9	9	9
11	20	60%	2	9	9	9	9
	21			10	10	10	10

	22	36%	3	8	8	8	8
	23	69%		12	12	12	12
	24	30%		17	18	18	17
	25	48%	4	10	10	10	10
	26	39%		16	8	8	8
	27	45%		11	8	8	8

Le tableau 5.10 montre que c'est la version AS_{rank} qui trouve les plus petits nombres de remaniements dans la plupart des cas. Ainsi, puisque nous privilégions la minimisation du nombre de remaniements, nous utiliserons, dans tout ce qui suit, la version AS_{rank} à chaque fois que nous ferons allusion à l'algorithme de colonie de fourmis.

Dans [79], Moussi avait utilisé la version MMAS de l'algorithme de colonie de fourmis. Cependant, il n'avait pas fait une comparaison entre les différentes versions. Une autre différence entre l'algorithme de colonie de fourmis qui est proposé dans [79] et ceux que nous proposons se trouve au niveau de la méthode de construction d'une solution. En effet, puisque l'objectif visé dans [79] était uniquement de minimiser la distance totale parcourue par les cavaliers gerbeurs (car les remaniements étaient complètement interdits), les fourmis construisaient des solutions en privilégiant les affectations qui minimisent les distances. Par contre, puisque nous minimisons les remaniements au lieu de les interdire (afin de prendre en considération le cas où leur interdiction rend le stockage impossible à réaliser), lors de la construction d'une solution, les fourmis commencent par exploiter les affectations qui ne causent pas de remaniements. Ces derniers n'utiliserons les autres possibilités d'affectation que s'ils sont dans l'incapacité d'affecter tous les conteneurs sans aucun remaniement. En plus de cette différence là, les algorithmes de colonie de fourmis que nous proposons ont chacun trois objectifs : la minimisation du nombre de remaniements, la minimisation de la distance totale à parcourir lors des chargements de conteneurs sur les moyens de transport, et la minimisation des distances qui séparent les emplacements de stockage de conteneurs d'une même catégorie.

5.6 Algorithme génétique

L'algorithme génétique que nous proposons pour la résolution du problème de stockage de conteneurs comprend principalement quatre étapes qui sont : la création d'une population initiale, la sélection d'une partie de cette population, l'application de croisements entre les individus sélectionnés, et la mutation d'un individu. Il est représenté par la figure suivante.

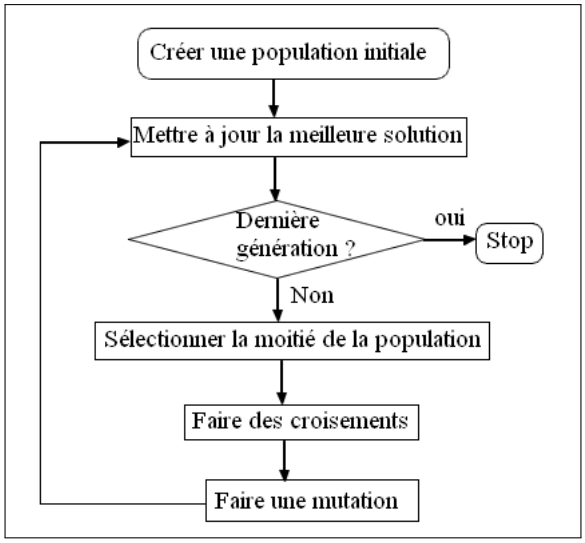


FIGURE 41 – Algorithme génétique

5.6.1 Création de la population initiale

Dans notre algorithme génétique, chaque individu de la population est une solution valide du problème de stockage de conteneurs. Par conséquent, nous représentons chaque individu sous forme d'un tableau qui a deux lignes et dont le nombre de colonnes est égal au nombre de conteneurs à stocker. Les conteneurs sont notés dans la première ligne alors que les piles sont notés dans la deuxième ligne (voir Figure 38). Les individus de la population initiale sont créés un par un. Le pseudo-code qui suit montre comment fonctionne la procédure de création d'un individu.

1. $a = 1$
2. $S = (s_{i,j})_{1 \leq i \leq 2,\ 1 \leq j \leq N}$ (c'est la solution en cours de création)
3. Tant que $(a \leq N)$ faire
   - 3.a. Choisir aléatoirement un conteneur k qui n'est pas encore traité
   - 3.b. Choisir aléatoirement une pile p qui n'est pas pleine, qui a la même taille que k, et qui respecte les contraintes d'ordre d'arrivée et de départ, c'est-à-dire qui ne contient pas de conteneur k' qui est tel que $k' > k$ ou $T_{k'} < T_k$
   - 3.c. $s_{1,a} = k,$ et $s_{2,a} = p$
   - 3.d. $a = a + 1$
4. Fin tant que

5.6.2 Méthodes de sélection

La sélection est la première phase de renouvellement d'une population. Comme son nom l'indique, elle permet de sélectionner les individus qui vont donner naissance à la descendance d'une génération. Ils existent différentes méthodes de sélection (voir la section 3.3.6), nous allons appliquer à notre algorithme génétique, séparément, deux parmi elles, à savoir : la roulette de la chance et la méthode élitiste.

5.6.3 Méthode de croisement

Dans un algorithme génétique, l'opérateur croisement sert à créer de nouveaux individus (appelés enfants), en combinant des parties de d'autres individus (appelés parents). Ils existent différentes méthodes de croisement (voir la section 3.3.6). Cependant, nous utilisons la méthode de croisement en un point, car le croisement en deux points ainsi que les autres méthodes nécessitent plus de corrections qui sont en rapport avec les contraintes de positionnement des conteneurs suivant l'ordre croissant de leurs dates d'arrivée et l'ordre décroissant de date de départ.

Lors d'un croisement, on choisit aléatoirement deux parents parmi les individus sélectionnés et un nombre réel (rc) compris entre 0 et 1. Si rc est supérieur à la probabilité de croisement (pc), alors on poursuit le croisement en sélectionnant aléatoirement un point de section. Après cela, on intervertit les parties des deux parents qui sont de part et d'autre du point de section. Ainsi on crée deux enfants. Ensuite, on vérifie s'ils sont valides, en d'autres termes s'ils satisfont toutes les contraintes. S'il y a des contraintes qui sont violées, on effectue les corrections nécessaires. La procédure de vérification et de correction d'un enfant S_j est la suivante.

1) Vérifier si les conteneurs qui sont affectés à chaque pile se succèdent suivant l'ordre croissant de leurs dates d'arrivée et l'ordre décroissant de leurs dates de départ, ensuite supprimer les affectations qui ne respectent pas cette règle.
Par exemple, si deux conteneurs k et k' sont affectés à la même pile et apparaissent dans S_j suivant le même ordre, si on a $k < k'$ et $T_k < T_{k'}$ alors on efface le contenu de la colonne de S_j qui contient k'.
2) Vérifier s'il y a des conteneurs qui sont présents dans deux colonnes différentes (c'est-à-dire des conteneurs qui sont affectés à deux emplacements différents). S'il y en a, alors garder la colonne (k, p) qui minimise $d_p^k + y_p^k + \sum_{k' \in S_j : k' > k} z_k^{k'}$
et ensuite supprimer l'autre.
3) Vérifier s'il y a des piles p qui ont trop de conteneurs. S'il y en a, alors supprimer les conteneurs k qui maximisent $d_p^k + y_p^k + \sum_{k' \in S_j : k' > k} z_k^{k'}$. Le nombre de conteneurs supprimés pour chaque pile doit être égal à son excédent.
4) Vérifier s'il y a des conteneurs qui sont absents. S'il y a en, alors chercher pour chacun d'entre eux la pile qui minimise $d_p^k + y_p^k + \sum_{k' > k} z_k^{k'}$ parmi celles qui sont compatibles et qui respectent les contraintes de capacité et d'ordre.
Utiliser les affectations obtenues pour remplir les colonnes vides de S_j.

Exemple

Supposons que l'on ait deux parents qui représentent chacun une affectation valide de 5 conteneurs dans un terminal qui dispose de 9 piles. Les opérations de croisement et de correction se déroulent comme suit.

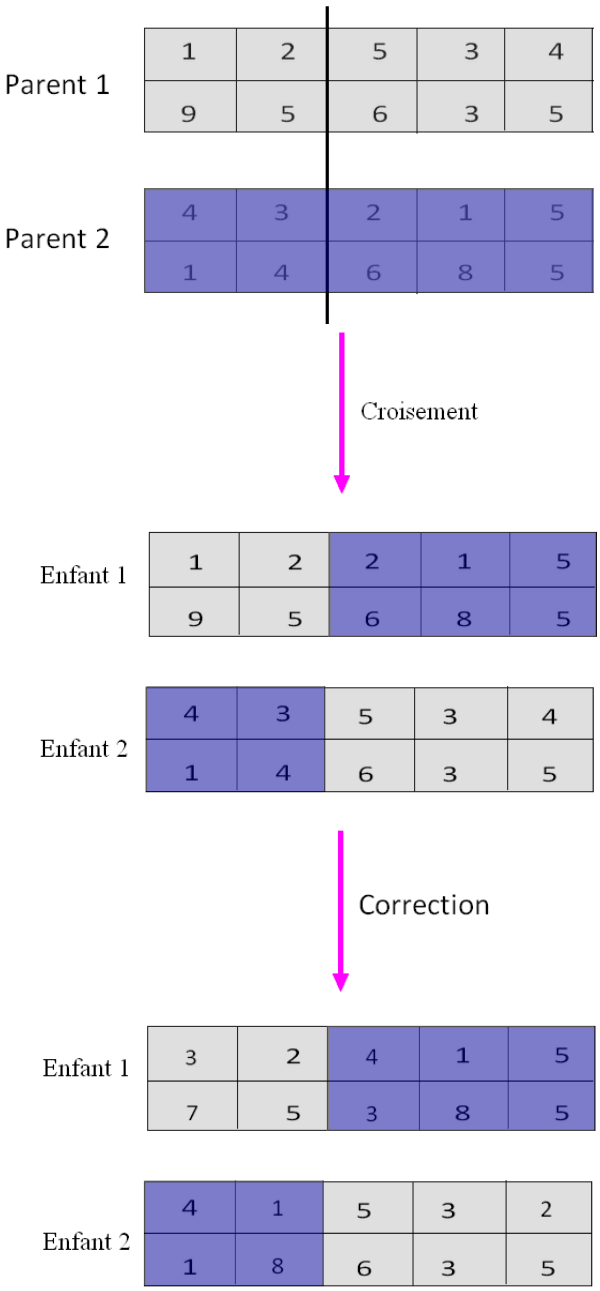


FIGURE 42 – Exemple de croisement

Dans cet exemple, on peut constater deux types d'anomalies dans les enfants : des conteneurs qui sont affectés à plusieurs emplacements (c'est le cas des conteneurs 1 et 2 dans l'enfant 1, et des conteneurs 3 et 4 dans l'enfant 2), et des conteneurs qui sont absents (c'est le cas des conteneurs 3 et 4 dans l'enfant 1, et des conteneurs 1 et 2 dans l'enfant 2). Rappelons que dans chaque solution, les conteneurs sont notés dans la première ligne, tandis que les piles sont notées dans la deuxième ligne.

5.6.4 Méthode de mutation

La mutation sert à diversifier les générations afin d'éviter une convergence prématurée vers un optimum local. Dans notre algorithme génétique, elle est effectuée à la fin de chaque itération en fonction d'une probabilité que l'on nomme pm. Ainsi, à la fin de chaque itération, on choisit aléatoirement entre 0 et 1 un nombre réel rm. Si $rm < pm$, alors on fait une mutation. Pour ce faire, on choisit aléatoirement un individu de la population et un conteneur que l'on affecte à une autre pile qui est elle aussi choisie aléatoirement parmi celles qui ne sont pas pleines et qui satisfont les contraintes de compatibilité, de capacité, et d'ordre.

5.6.5 Résultats numériques

Dans cette section, nous comparons les résultats fournis par les deux variantes d'algorithme génétique que nous proposons (à savoir : la méthode de sélection basée sur la roue de la fortune, et celle qui est élitiste). Pour ce faire, nous commençons d'abord par chercher les bonnes valeurs des paramètres utilisés dans ces algorithmes. Les résultats obtenus sont marqués dans le tableau 5.12. Chaque paramètre est traité individuellement, en fixant les autres paramètres, et en le faisant varier.

Tableau 5.12 – Valeurs des paramètres de l'algorithme génétique

Paramètre	Valeur
Nombre de générations	$NG_{Max} = 42$
Taille de la population	$NI_{Max} = 45$
Probabilité de croisement	$pc = 0.75$
Probabilité de mutation	$pm = 0.25$

Pour fixer le nombre de générations, nous avons considéré les nombres entiers compris entre 20 et 100, et nous avons choisi celui à partir duquel la valeur de la fonction objectif ne diminue plus. De façon similaire, la recherche de la taille adéquate d'une population a été faite en considérant les nombres entiers compris entre 10 et 100. Quant aux probabilités de croisement et de mutation, ils ont été fixés en considérant les nombres réels compris entre 0 et 1, avec un pas de 0,05.

Après avoir fixé les valeurs des paramètres, nous comparons la méthode de sélection élitiste à celle de la roulette de la chance. Pour ce faire, nous utilisons les mêmes instances qui ont été utilisées pour comparer les différentes versions d'algorithme de colonie de fourmis. Les résultats obtenus sont notés dans les tableaux 5.13 et 5.14.

Tableau 5.13: Résultats numériques de l'algorithme génétique avec la méthode de la roulette de la chance

Instances				CLPEX				Génétique-Roulette				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	26913	0 sec	16855	10	0.16739
	21	53%		23874	3.12 sec	16731	6	29208	1 sec	17130	12	0.223423
	22	43%	3	28863	17.61 sec	17724	8	32564	0 sec	18529	14	0.128226
	23	46%		23751	9.78 sec	18593	1	28841	0 sec	18803	10	0.214307
	24	70%		22529	195.91 sec	19310	3	34083	0 sec	19021	15	0.51285
	25	63%	4	21331	63.73 sec	20145	1	26879	1 sec	20854	6	0.260091
	26	36%		26962	225.02 sec	20831	3	32741	0 sec	20701	12	0.214339
	27	36%		23107	2.11 sec	21951	1	31119	0 sec	22083	9	0.346735
	28	53%		24965	341.53 sec	22700	1	33079	2 sec	23004	10	0.325015
11	20	60%	2	18129	18.45 sec	16048	2	23293	0 sec	16240	7	0.284847
	21			20823	1393.86 sec	16682	4	28228	0 sec	17172	11	0.355616
	22	36%	3	24131	155.91 sec	18040	3	31758	0 sec	17690	14	0.316066
	23	69%		19490	19.78 sec	18360	1	28835	0 sec	18753	10	0.479477
	24	30%		32568	364.96 sec	19383	11	36018	0 sec	19941	16	0.105932
	25	48%	4	22294	17.39 sec	20103	1	27909	0 sec	19842	8	0.251861
	26	39%		23801	38.39 sec	20646	2	30181	1 sec	21116	9	0.268056
	27	45%		25252	1049.26 sec	22076	1	31779	0 sec	21716	10	0.258475
Total				404924	3920.19 sec	325261	55	513428	5 sec	329450	183	4.71271

Les résultats du tableau 5.13 montrent que l'écart relatif moyen entre les résultats fournis par l'algorithme génétique dans lequel on utilise la roulette de la chance comme méthode de sélection et les résultats optimaux trouvés par "ILGO CPLEX" est égal à 0.27722.

Tableau 5.14: Résultats numériques de l'algorithme génétique avec la méthode élitiste

Instances				CLPEX				Génétique-Élitiste				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	26305	1 sec	16253	10	0.141017
	21	53%		23874	3.12 sec	16731	6	28166	9 sec	17099	11	0.179777
	22	43%	3	28863	17.61 sec	17724	8	31705	0 sec	17676	14	0.0984652
	23	46%		23751	9.78 sec	18593	1	31400	0 sec	18359	13	0.32205
	24	70%		22529	195.91 sec	19310	3	30800	6 sec	18765	12	0.367127
	25	63%	4	21331	63.73 sec	20145	1	23563	5 sec	20547	3	0.104636
	26	36%		26962	225.02 sec	20831	3	33588	7 sec	20544	13	0.245753
	27	36%		23107	2.11 sec	21951	1	33613	8 sec	21583	12	0.454667
	28	53%		24965	341.53 sec	22700	1	31732	0 sec	22687	9	0.271059
11	20	60%	2	18129	18.45 sec	16048	2	25228	1 sec	16178	9	0.391583
	21	60%		20823	1393.86 sec	16682	4	26856	7 sec	16804	10	0.289728
	22	36%	3	24131	155.91 sec	18040	3	31687	0 sec	17624	14	0.313124
	23	69%		19490	19.78 sec	18360	1	30476	0 sec	18413	12	0.563674

	24	30%		32568	364.96 sec	19383	11	35447	2 sec	19381	16	0.0883997
	25	48%		22294	17.39 sec	20103	1	30853	0 sec	19798	11	0.383915
	26	39%	4	23801	38.39 sec	20646	2	28902	2 sec	20865	8	0.214319
	27	45%		25252	1049.26 sec	22076	1	29879	1 sec	21818	8	0.183233
Total				404924	3920.19 sec	325261	55	510200	49 sec	324394	185	4.61253

Les résultats du tableau 5.14 montrent que l'écart relatif moyen entre les résultats fournis par l'algorithme génétique dans lequel on utilise la méthode de sélection élitiste et les résultats optimaux trouvés par "ILGO CPLEX" est égal à 0.27133.

La comparaison des écarts associés à la méthode élitiste et à celle de la roulette de la chance montre que c'est la méthode élitiste qui donne en moyenne les résultats les plus proches des optimums. Une comparaison des valeurs des solutions trouvées par ces deux méthodes est donnée dans le tableau 5.15.

Tableau 5.15: Comparaison des valeurs de la fonction objectif

Instances				Génétique-Roulette	Génétique-Élitiste
Np	N	P_d	H		
10	20	33%	2	26913	26305
	21	53%		29208	28166
	22	43%	3	32564	31705
	23	46%		28841	31400
	24	70%		34083	30800
	25	63%	4	26879	23563
	26	36%		32741	33588
	27	36%		31119	33613
	28	53%		33079	31732
11	20	60%	2	23293	25228
	21			28228	26856
	22	36%	3	31758	31687
	23	69%		28835	30476
	24	30%		36018	35447
	25	48%	4	27909	30853
	26	39%		30181	28902
	27	45%		31779	29879

À partir du tableau 5.15, on peut constater que la méthode élitiste donne les meilleurs résultats dans la plupart des cas.

Tableau 5.16: Comparaison des distances

Instances				Génétique-Roulette	Génétique-Élitiste
Np	N	P_d	H		
	20	33%	2	16855	16253

	21	53%		17130	17099
	22	43%	3	18529	17676
	23	46%		18803	18359
	24	70%		19021	18765
10	25	63%	4	20854	20547
	26	36%		20701	20544
	27	36%		22083	21583
	28	53%		23004	22687
	20	60%	2	16240	16178
	21			17172	16804
	22	36%	3	17690	17624
11	23	69%		18753	18413
	24	30%		19941	19381
	25	48%	4	19842	19798
	26	39%		21116	20865
	27	45%		21716	21818

Le tableau 5.16 montre que la méthode élitiste trouve les plus petites distances, dans la plupart des cas.

Tableau 5.17: Comparaison des nombres de remaniements

Instances				Génétique-Roulette	Génétique-Élitiste
Np	N	P_d	H		
	20	33%	2	10	10
	21	53%		12	11
	22	43%	3	14	14
	23	46%		10	13
10	24	70%		15	12
	25	63%	4	6	3
	26	36%		12	13
	27	36%		9	12
	28	53%		10	9
	20	60%	2	7	9
	21			11	10
	22	36%	3	14	14
11	23	69%		10	12
	24	30%		16	16
	25	48%	4	8	11
	26	39%		9	8
	27	45%		10	8

Les résultats contenus dans le tableau 5.17 montrent que la méthode élitiste trouve les plus petits nombres de remaniements dans la plupart des cas.

Après ces comparaisons, on peut dire que la méthode élitiste est préférable à la méthode de la roulette de la chance pour la résolution du problème de stockage de conteneurs.

Nous l'utilisons donc, dans ce qui suit, à chaque fois que nous faisons allusion à l'algorithme génétique.

Dans [79], Moussi avait proposé un algorithme génétique dans lequel la méthode de la roulette de la chance était utilisée pour effectuer des sélections. Mis à part le fait qu'il minimisait la distance totale parcourue par les cavaliers gerbeurs dans un terminal à conteneurs où les remaniements sont complètement interdits, ils existent essentiellement des différences entre : nos méthodes de représentation d'un individu (solution), nos méthodes de création d'un individu, et nos méthodes de correction des enfants. En effet, dans [79], chaque individu est représenté sous forme d'un tableau qui a deux lignes et dont le nombre de colonnes est égal au nombre total d'emplacements (occupés et inoccupés). Alors que dans nos algorithmes génétiques, le nombre de colonnes de chaque individu est égal au nombre de conteneurs à stocker. Cela permet d'économiser de l'espace mémoire d'ordinateur, afin de faciliter la résolution d'instances de grandes tailles. En ce qui concerne la méthode de création d'un individu, dans [79] les conteneurs sont traités dans l'ordre lexicographique, en affectant chacun à un emplacement de stockage convenable qui est choisi aléatoirement. Contrairement à cette façon de faire, nous construisons un individu en traitant les conteneurs dans un ordre aléatoire, en affectant chacun à un emplacement de stockage convenable qui est choisi aléatoirement. Cela nous permet d'avoir une population initiale très diversifiée. Quant à la méthode de correction d'un enfant obtenu après un croisement, dans [79] les doubles affections d'un conteneur sont corrigées simplement en supprimant la deuxième affectation. Tandis que dans nos algorithmes, nous corrigeons les doubles affectations en conservant l'affectation qui minimise la fonction objectif, et en éliminant l'autre. Similairement, dans [79], les conteneurs qui ne sont pas présents dans un enfant obtenu suite à un croisement sont ajoutés individuellement, en testant les piles dans l'ordre lexicographique et en affectant chaque conteneur à la première pile où il peut être stocké sans qu'aucune contrainte ne soit violée. Le problème avec cette méthode de correction est le fait que les dernières piles ne sont sollicitées que lorsque les premières sont pleines. Cela ne garantit pas une bonne exploitation de l'espace de recherche de solution. Ainsi pour améliorer cela, nous affectons chacun des conteneurs manquants à la pile qui minimise la fonction objectif.

5.7 Algorithme hybride de colonie de fourmis et génétique (HCFG)

Cet algorithme est une combinaison de l'algorithme de colonie de fourmis et de l'algorithme génétique. Par conséquent, il contient toutes les procédures de ces deux algorithmes. Cependant, sa principale différence avec l'algorithme génétique est le fait qu'une partie de la population initiale est construite par des fourmis alors que l'autre partie est construite de la même façon qu'avec l'algorithme génétique. L'algorithme HCFG commence par la création des couples et l'initialisation des traces de phéromone. Après cela, chaque fourmi construit une solution. On obtient ainsi un nombre d'individus auxquels on ajoute d'autres solutions pour avoir une population initiale sur laquelle on applique l'algorithme génétique. Par la suite, on met à jour les traces de phéromone, ensuite on recommence une nouvelle itération. Ces étapes sont répétées jusqu'à l'itération finale,

comme on peut le constater sur la figure 43, où les étapes colorées en vert font référence à l'algorithme de colonie de fourmis, tandis que celles colorées en rose font parties de l'algorithme génétique.

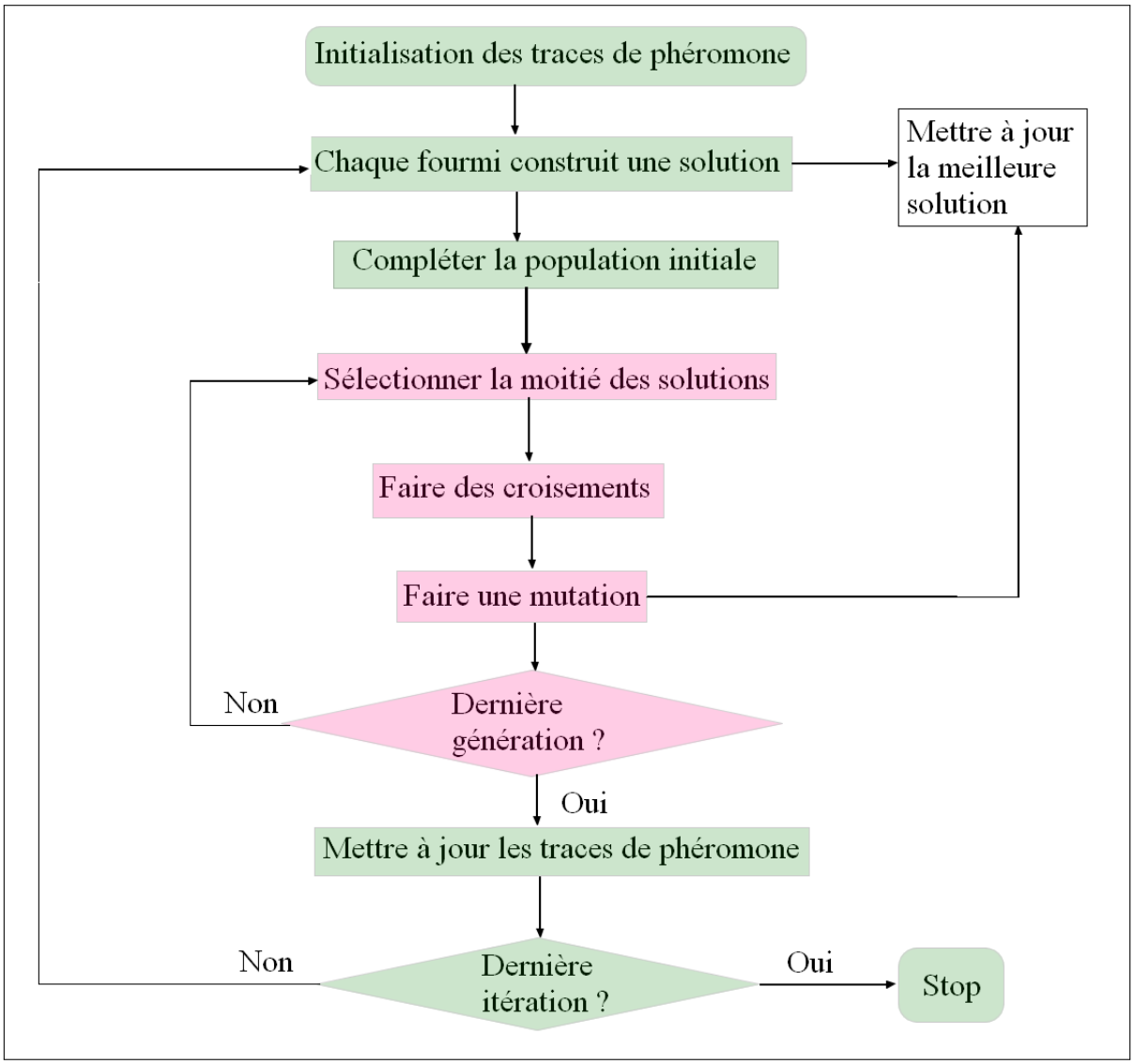


FIGURE 43 – Algorithme hybride de colonie de fourmis et génétique

5.7.1 Résultats Numériques

Lors de l'implémentation de l'algorithme HCFG, nous avons utilisé les mêmes valeurs des paramètres qui sont utilisés dans l'algorithme génétique et dans l'algorithme de colonie de fourmis. Ils sont résumés dans le tableau 5.18 suivant.

Tableau 5.18: Valeurs des paramètres de l'algorithme HCFG

Paramètre	Valeur
Nombre d'itérations	$NT_{Max} = 40$
Nombre de fourmis	$NA = 17$
Nombre de générations	$NG_{Max} = 42$
Taille d'une population	$NI_{Max} = 45$
Probabilité de croisement	$pc = 0.75$
Probabilité de mutation	$pm = 0.25$
Puissance de la phéromone	$\alpha = 0.3$
Puissance de la visibilité	$\beta = 0.9$
Taux d'évaporation de la phéromone	$\rho = 0.2$
Nombre de solutions sélectionnées	$w = 8$
Bonus de phéromone	$e = 8$

Pour évaluer la qualité des solutions fournies par l'algorithme HCFG, nous les avons comparé aux solutions optimales fournies par "ILOG CPLEX". Les résultats sont notés dans le tableau 5.19.

Tableau 5.19: Résultats numériques de l'algorithme HCFG

Instances				CLPEX				HCFG				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	25303	5 sec	16251	9	0.0975536
	21	53%		23874	3.12 sec	16731	6	27164	18 sec	17097	10	0.137807
	22	43%	3	28863	17.61 sec	17724	8	29468	1 sec	18431	11	0.0209611
	23	46%		23751	9.78 sec	18593	1	26280	10 sec	19228	7	0.10648
	24	70%		22529	195.91 sec	19310	3	29798	12 sec	18762	11	0.322651
	25	63%	4	21331	63.73 sec	20145	1	22559	15 sec	20547	2	0.0575688
	26	36%		26962	225.02 sec	20831	3	27555	2 sec	21519	6	0.0219939
	27	36%		23107	2.11 sec	21951	1	26069	7 sec	22039	4	0.128186
	28	53%		24965	341.53 sec	22700	1	29998	3 sec	23929	6	0.201602
	20	60%	2	18129	18.45 sec	16048	2	23956	8 sec	15929	1	0.321419
	21			20823	1393.86 sec	16682	4	25817	4 sec	16783	9	0.239831

11	22	36%	3	24131	155.91 sec	18040	3	24990	0 sec	17902	7	0.0355974
	23	69%		19490	19.78 sec	18360	1	29317	1 sec	19262	10	0.504207
	24	30%		32568	364.96 sec	19383	11	32847	2 sec	19757	13	0.00856669
	25	48%	4	22294	17.39 sec	20103	1	29864	1 sec	20791	9	0.339553
	26	39%		23801	38.39 sec	20646	2	28452	2 sec	21407	7	0.195412
	27	45%		25252	1049.26 sec	22076	1	27916	9 sec	22853	5	0.105497
Total				404924	3920.19 sec	325261	55	467353	100 sec	332487	127	2.84489

À partir du tableau 5.19, on peut constater que l'écart relatif moyen entre les résultats de HCFG et les résultats optimaux trouvés par "ILOG CPLEX" est égal à 0.167346.

Des comparaisons entre l'algorithme HCFG, l'algorithme de colonie de fourmis, et l'algorithme génétique sont faites dans les tableaux 5.20, 5.21, et 5.22.

Tableau 5.20: Comparaison des valeurs de la fonction objectif

Instances				HCFG	Fourmi	Génétique
Np	N	P_d	H			
10	20	33%	2	25303	27524	26305
	21	53%		27164	28100	28166
	22	43%	3	29468	31869	31705
	23	46%		26280	28741	31400
	24	70%		29798	33946	30800
	25	63%	4	22559	28375	23563
	26	36%		27555	33735	33588
	27	36%		26069	27071	33613
	28	53%		29998	31142	31732
11	20	60%	2	23956	24958	25228
	21			25817	26819	26856
	22	36%	3	24990	25992	31687
	23	69%		29317	30271	30476
	24	30%		32847	36332	35447
	25	48%	4	29864	29984	30853
	26	39%		28452	29072	28902
	27	45%		27916	30053	29879

Le tableau 5.20 montre que l'algorithme HCFG améliore les résultats de l'algorithme génétique et ceux de l'algorithme de colonie de fourmis.

Tableau 5.21: Comparaison des distances

Instances				HCFG	Fourmi	Génétique
Np	N	P_d	H			
10	20	33%	2	16251	17463	16253
	21	53%		17097	17055	17099
	22	43%	3	18431	17849	17676

	23	46%		19228	18705	18359
	24	70%		18762	18930	18765
	25	63%	4	20547	20333	20547
	26	36%		21519	20712	20544
	27	36%		22039	22041	21583
	28	53%		23929	22066	22687
11	20	60%	2	15929	15931	16178
	21			16783	16785	16804
	22	36%	3	17902	17908	17624
	23	69%		19262	18182	18413
	24	30%		19757	19280	19381
	25	48%	4	20791	19964	19798
	26	39%		21407	21045	20865
	27	45%		22853	22002	21818

Le tableau 5.21 montre que l'algorithme génétique trouve les plus petites distances dans la plupart des cas, comparé à l'algorithme de colonie de fourmis et à l'algorithme HCFG.

Tableau 5.22: Comparaison des nombres de remaniements

Instances				HCFG	Fourmi	Génétique
Np	N	P_d	H			
10	20	33%	2	9	10	10
	21	53%		10	11	11
	22	43%	3	11	14	14
	23	46%		7	10	13
	24	70%		11	15	12
	25	63%	4	2	8	3
	26	36%		6	13	13
	27	36%		4	5	12
	28	53%		6	9	9
11	20	60%	2	1	9	9
	21			9	10	10
	22	36%	3	7	8	14
	23	69%		10	12	12
	24	30%		13	17	16
	25	48%	4	9	10	11
	26	39%		7	8	8
	27	45%		5	8	8

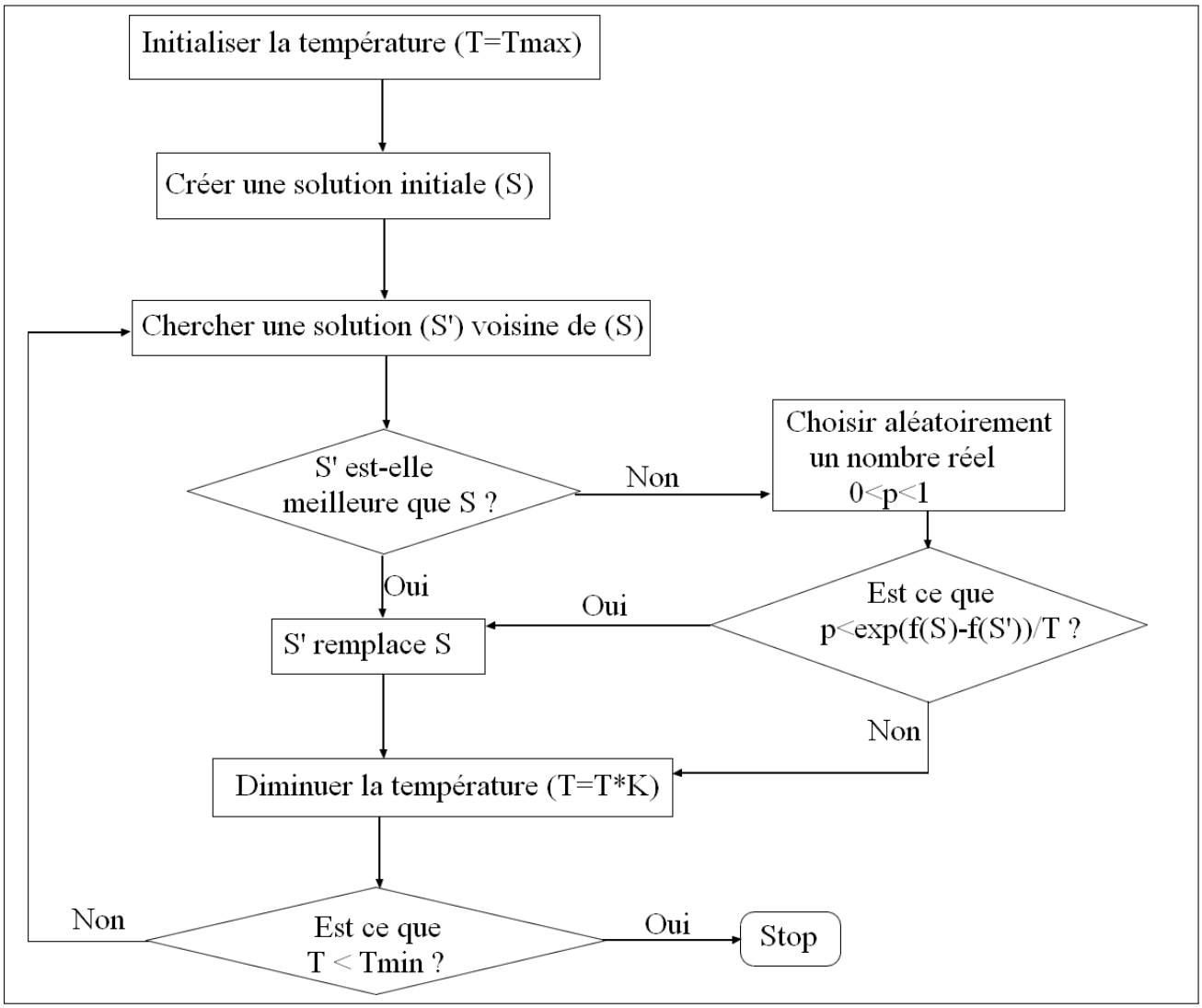
Le tableau 5.22 montre que l'algorithme HCFG trouve les plus petits nombres de remaniements dans la plupart des cas, comparé à l'algorithme génétique et à l'algorithme de colonie de fourmis.

Au vu de ces résultats, et au fait que nous privilégions la minimisation des nombres

de remaniements, nous pouvons dire que l'hybridation HCFG permet d'améliorer les résultats de l'algorithme génétique et ceux de l'algorithme de colonie de fourmis.

5.8 Algorithme du recuit simulé (RS)

Le recuit simulé est un algorithme méta-heuristique qui fut créé en 1983 par Kirkpatrick et al. [65]. Ces derniers se sont inspirés d'un processus métallurgique qui a pour objectif d'améliorer l'aspect d'un métal, en le chauffant jusqu'à une température très élevée puis en le refroidissant graduellement. L'algorithme du recuit simulé fait partie de la classe des algorithmes de recherche locale, il explore le voisinage d'une solution initiale afin de trouver de meilleures solutions comme le montre la figure 44.



f est la fonction à optimiser, et K est un nombre réel compris entre 0 et 1

FIGURE 44 – Algorithme du recuit simulé

Le pseudo-code correspondant à cet organigramme est le suivant.

1. Créer une solution initiale (S), notons $f(S)$ la valeur de S
2. Initialiser la température ($T = T_{max}$)
3. Tant que ($T < T_{min}$) faire
    3.a. Rechercher une solution (S') voisine de S
    3.b. Si ($f(S') < f(S)$) alors
        3.b.1. S' remplace S
    3.c. Sinon
        3.c.1. Choisir aléatoirement un nombre réel p compris entre 0 et 1
        3.c.2. Si ($p < e^{\frac{f(S)-f(S')}{T}}$) alors
            3.c.2.a. S' remplace S
        3.c.3. Fin Si
    3.d. Fin Sinon
    3.e. $T = K \times T$
4. Fin Tant que

5.8.1 Création d'une solution initiale

Dans l'algorithme du recuit simulé que nous proposons pour la résolution du problème de stockage de conteneurs, nous représentons une solution sous la forme d'un tableau qui a deux lignes et dont le nombre de colonnes est égal aux nombre de conteneurs à stocker (voir figure 38). Les conteneurs sont notés dans la première ligne alors que les piles sont marquées dans la deuxième ligne.

Lors de la construction d'une solution initiale, nous affectons les conteneurs aux piles séquentiellement dans un ordre aléatoire comme le montre le pseudo-code suivant.

1. $a = 1$
2. $S = (s_{i,j})_{1 \leq i \leq 2, 1 \leq j \leq N}$ (c'est la solution en cours de création)
3. Tant que ($a \leq N$) faire
    3.a. Choisir aléatoirement un conteneur k qui n'est pas encore traité
    3.b. Choisir aléatoirement une pile p qui n'est pas pleine, qui a la même taille que k, et qui respecte les contraintes d'ordre d'arrivée et de départ, c'est-à-dire qui ne contient pas de conteneur k' qui est tel que $k' > k$ ou $T_{k'} < T_k$
    3.c. $s_{1,a} = k$, et $s_{2,a} = p$
    3.d. $a = a + 1$
4. Fin tant que

5.8.2 Recherche de solution voisine

Lors de la recherche d'une solution voisine, nous choisissions aléatoirement un conteneur que nous affectons à une autre pile comme le montre le pseudo-code suivant.

Tableau 5.25: Recherche locale

1. Fin = faux ;
2. Compteur = 1 ;

3. Tant que (Fin = faux, et Compteur $\leq N$) faire
  3.a. Sélectionner aléatoirement un conteneur k
  3.b. Soit p la pille affectée au conteneur k dans la solution courante
  3.c. Construire la liste L des piles qui ne sont pas pleines, qui ont la même taille que k, et qui vérifient les contraintes d'ordre
  3.d. Changement = faux
  3.e. j=1
  3.f. Tant que (Changement = faux, et $j \leq card(L)$) faire
    3.f.1. $p' = L(j)$
    3.f.2. Si $(d^k_{p'} + y^k_{p'} + A < d^k_p + y^k_p + B)$ alors
      3.f.2.a. Changement = vrai
    3.f.3. Sion
      3.f.3.a. $j = j + 1$
  3.g. Fin tant que
  3.h. Si (Changement = vrai) alors
    3.h.a. Affecter k à p'
    3.h.b. Mettre à jour les capacités des piles ($c_p = c_p + 1,$ et $c_{p'} = c_{p'} - 1$)
    3.h.b. Fin = vrai
  3.i. Sinon
  3.j. Compteur = Compteur +1
4. Fin tant que

$A = \sum_{k'>k}^{N} z_k^{k'}$ sachant que k est affecté à p'
$B = \sum_{k'>k}^{N} z_k^{k'}$ sachant que k est affecté à p

5.8.3 Résultats numériques

La première étape des résultats numériques consistait à chercher les bonnes valeurs des paramètres. Pour ce faire, nous avons traité les paramètres un par un. À chaque étape, on fait varier un paramètre, et on fixe les autres. La température maximale a été traitée, en considérant les nombres entiers compris entre 100 et 400 par pas de 10, et en choisissant la valeur à partir de laquelle la valeur de la fonction objectif ne diminue plus. La recherche de la température minimale a été faite de façon similaire en considérant les nombres entiers compris entre 1 et 50. Quant au coefficient de diminution de la température, il a été recherché dans l'intervalle [0.6, 0.999] par pas de 0.001. Les résultats obtenus sont notés dans le tableau 5.26.

Tableau 5.26 – Valeurs des paramètres de l'algorithme du recuit simulé

Paramètre	Valeur
Température maximale	$TE_{Max} = 290$
Température minimale	$TE_{Min} = 1$
Coefficient de diminution de la température	$K = 0.989$

Une comparaison des résultats de l'algorithme du recuit simulé par rapport aux ré-

sultats optimaux trouvés par "ILOG CPLEX" est présentée dans le tableau 5.27 suivant.

Tableau 5.27: Résultats numériques de l'algorithme du recuit simulé

Instances				CLPEX				Recuit simulé				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	gap
10	20	33%	2	23054	3.38 sec	15938	6	27524	0 sec	17463	10	0.193893
	21	53%		23874	3.12 sec	16731	6	30826	0 sec	17756	13	0.291195
	22	43%	3	28863	17.61 sec	17724	8	33555	0 sec	19515	14	0.162561
	23	46%		23751	9.78 sec	18593	1	31986	0 sec	18931	13	0.346722
	24	70%		22529	195.91 sec	19310	3	33683	0 sec	18635	15	0.495095
	25	63%	4	21331	63.73 sec	20145	1	25892	0 sec	20871	5	0.21382
	26	36%		26962	225.02 sec	20831	3	35786	0 sec	20755	15	0.327275
	27	36%		23107	2.11 sec	21951	1	34903	0 sec	21876	13	0.510495
	28	53%		24965	341.53 sec	22700	1	32260	0 sec	23174	9	0.292209
11	20	60%	2	18129	18.45 sec	16048	2	25362	0 sec	16312	9	0.398974
	21			20823	1393.86 sec	16682	4	28018	0 sec	16972	11	0.345531
	22	36%	3	24131	155.91 sec	18040	3	31859	0 sec	17793	14	0.320252
	23	69%		19490	19.78 sec	18360	1	32340	0 sec	18274	14	0.659312
	24	30%		32568	364.96 sec	19383	11	37198	0 sec	20109	17	0.142164
	25	48%	4	22294	17.39 sec	20103	1	30757	0 sec	19696	11	0.379609
	26	39%		23801	38.39 sec	20646	2	30752	0 sec	20710	10	0.292047
	27	45%		25252	1049.26 sec	22076	1	34155	0 sec	22087	12	0.352566
Total				404924	3920.19 sec	325261	55	536856	0 sec	330929	205	5.72372

À partir du tableau 5.27, on peut constater que l'écart relatif moyen entre les résultats du recuit simulé et les résultats optimaux trouvés par "ILOG CPLEX" est de 0.33669.

Mis à part les objectifs à minimiser, les méthodes de codage et de création de solution, notre algorithme de recuit simulé diffère aussi de celui qui a été proposé dans [79] par la façon de rechercher une voisine d'une solution. En effet, dans [79], la recherche locale se faisait uniquement en choisissant aléatoirement un conteneur, et en l'affectant à une autre pile convenable qui est elle aussi choisie aléatoirement. Alors que dans notre algorithme de recuit simulé, la recherche locale est faite en choisissant aléatoirement un conteneur, et en l'affectant à la meilleure pile parmi celles qui lui conviennent.

5.9 Algorithme hybride de colonie de fourmis et recuit simulé (HCFRS)

Cette hybridation est un renforcement de l'algorithme de colonie de fourmis par le recuit simulé. En effet, c'est un algorithme de colonie de fourmis dans lequel le recuit simulé est utilisé en tant qu'une méthode de recherche locale pour améliorer les solutions construites par les fourmis à la fin de chaque itération, comme le montre la figure 45.

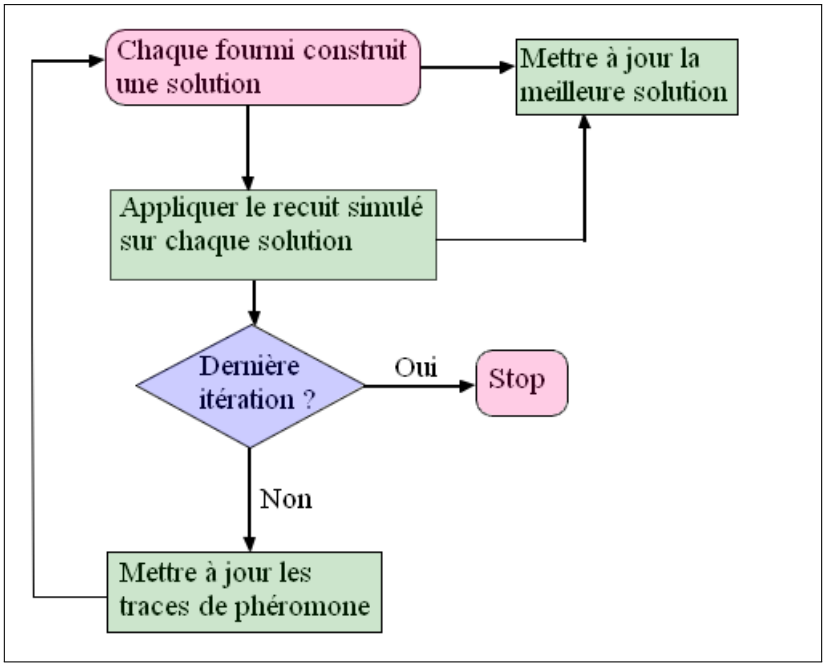


FIGURE 45 – Algorithme hybride de colonie de fourmis et de recuit simulé

L'algorithme HCFRS commence par l'initialisation des traces de phéromone. Après cela, chaque fourmi construit une solution. Ensuite, on applique l'algorithme du recuit simulé sur chacune de ces solutions. Puis on met à jour les traces de phéromone, et on recommence une nouvelle itération.

5.9.1 Résultats Numériques

Pour implémenter l'algorithme HCFRS, nous avons utilisé les mêmes valeurs des paramètres qui sont utilisés dans l'algorithme de colonie de fourmis et dans l'algorithme du recuit simulé. Une comparaison des résultats de HCFRS par rapport aux résultats optimaux trouvés par "ILOG CPLEXX" est donnée dans le tableau 5.28.

Tableau 5.28: Résultats numériques de l'algorithme HCFRS

Instances				CLPEX				HCFRS				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	25303	5 sec	16251	9	0.0975536
	21	53%		23874	3.12 sec	16731	6	27164	18 sec	17097	10	0.137807
	22	43%	3	28863	17.61 sec	17724	8	29468	1 sec	18431	11	0.0209611
	23	46%		23751	9.78 sec	18593	1	26280	10 sec	19228	7	0.10648
	24	70%		22529	195.91 sec	19310	3	29798	12 sec	18762	11	0.322651
	25	63%	4	21331	63.73 sec	20145	1	22559	15 sec	20547	2	0.0575688
	26	36%		26962	225.02 sec	20831	3	27555	2 sec	21519	6	0.0219939
	27			23107	2.11 sec	21951	1	26069	7 sec	22039	4	0.128186
	28	53%		24965	341.53 sec	22700	1	29998	3 sec	23929	6	0.201602

11	20	60%	2	18129	18.45 sec	16048	2	23956	8 sec	15929	1	0.321419
	21			20823	1393.86 sec	16682	4	25817	4 sec	16783	9	0.239831
	22	36%	3	24131	155.91 sec	18040	3	24990	0 sec	17902	7	0.0355974
	23	69%		19490	19.78 sec	18360	1	29317	1 sec	19262	10	0.504207
	24	30%		32568	364.96 sec	19383	11	32847	2 sec	19757	13	0.00856669
	25	48%	4	22294	17.39 sec	20103	1	29864	1 sec	20791	9	0.339553
	26	39%		23801	38.39 sec	20646	2	28452	2 sec	21407	7	0.195412
	27	45%		25252	1049.26 sec	22076	1	27916	9 sec	22853	5	0.105497
Total				404924	3920.19	325261	55	467353	100	332487	127	2.84489

À partir des résultats contenus dans la tableau 5.28, on peut remarquer que l'écart relatif moyen entre les résultats de l'algorithme HCFRS et les résultats optimaux trouvés par "ILOG CPLEX" est de 0.167347.

Des comparaisons entre l'algorithme HCFRS, l'algorithme de colonie de fourmis, et l'algorithme du recuit simulé sont faites dans les tableaux 5.29, 5.30, et 5.32.

Tableau 5.29: Comparaison des valeurs de la fonction objectif

Instances				HCFRS	Fourmis	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	26522	27524	27524
	21	53%		27098	28100	30826
	22	43%	3	28680	31869	33555
	23	46%		26255	28741	31986
	24	70%		30878	33946	33683
	25	63%	4	23431	28375	25892
	26	36%		32759	33735	35786
	27			26069	27071	34903
	28	53%		28497	31142	32260
11	20	60%	2	22064	24958	25362
	21			26711	26819	28018
	22	36%	3	24990	25992	31859
	23	69%		28515	30271	32340
	24	30%		35856	36332	37198
	25	48%	4	26362	29984	30757
	26	39%		22889	29072	30752
	27	45%		25343	30053	34155

Le tableau 5.29 montre que l'algorithme HCFRS améliore les résultats de l'algorithme du recuit simulé et ceux de l'algorithme de colonie de fourmis.

Tableau 5.30: Comparaison des distances

Instances				HCFRS	Fourmis	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	17461	17463	17463
	21	53%		17053	17055	17756
	22	43%	3	17638	17849	19515
	23	46%		19212	18705	18931
	24	70%		18848	18930	18635
	25	63%	4	20403	20333	20871
	26	36%		20731	20712	20755
	27			22039	22041	21876
	28	53%		23429	22066	23174
11	20	60%	2	16013	15931	16312
	21			16678	16785	16972
	22	36%	3	17906	17908	17793
	23	69%		18441	18182	18274
	24	30%		19809	19280	20109
	25	48%	4	19326	19964	19696
	26	39%		20858	21045	20710
	27	45%		21260	22002	22087

Le tableau 5.30 montre que l'algorithme de colonie de fourmis donne les plus petites distances dans la plupart des cas, comparé à l'algorithme HCFRS et à l'algorithme du recuit simulé.

Tableau 5.31: Comparaison des nombres de remaniements

Instances				HCFRS	Fourmis	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	9	10	10
	21	53%		10	11	13
	22	43%	3	11	14	14
	23	46%		7	10	13
	24	70%		12	15	15
	25	63%	4	3	8	5
	26	36%		12	13	15
	27			4	5	13
	28	53%		5	9	9
11	20	60%	2	6	9	9
	21			10	10	11
	22	36%	3	7	8	14
	23	69%		10	12	14
	24	30%		16	17	17
	25	48%	4	7	10	11
	26	39%		2	8	10

	27	45%		4	8	12

Le tableau 5.31 montre que l'algorithme HCFRS donne les plus petits nombres de remaniements dans la plupart des cas, comparé à l'algorithme de colonie de fourmis et à l'algorithme du recuit simulé.

Au vu de ces résultats, et en tenant compte du fait que l'on privilégie la minimisation du nombre de remaniements, on peut dire que l'hybridation HCFRS améliore l'algorithme de colonie de fourmis et l'algorithme du recuit simulé.

5.10 Algorithme hybride génétique et recuit simulé (HGRS)

L'algorithme HGRS est une hybridation qui permet de renforcer l'algorithme génétique grâce au recuit simulé. Ainsi, le recuit simulé est appliqué sur chaque individu (qu'il fasse partie de la population initiale ou bien des autres générations). Comme pour l'algorithme génétique, l'algorithme HGRS commence par la création d'une population initiale. Ensuite, on applique le recuit simulé sur tous les individus de la population initiale. Puis, on met à jour la meilleure solution. Après cela, on effectue les opérations de croisement et de mutation, et on recommence une nouvelle itération, comme le montre la figure 46.

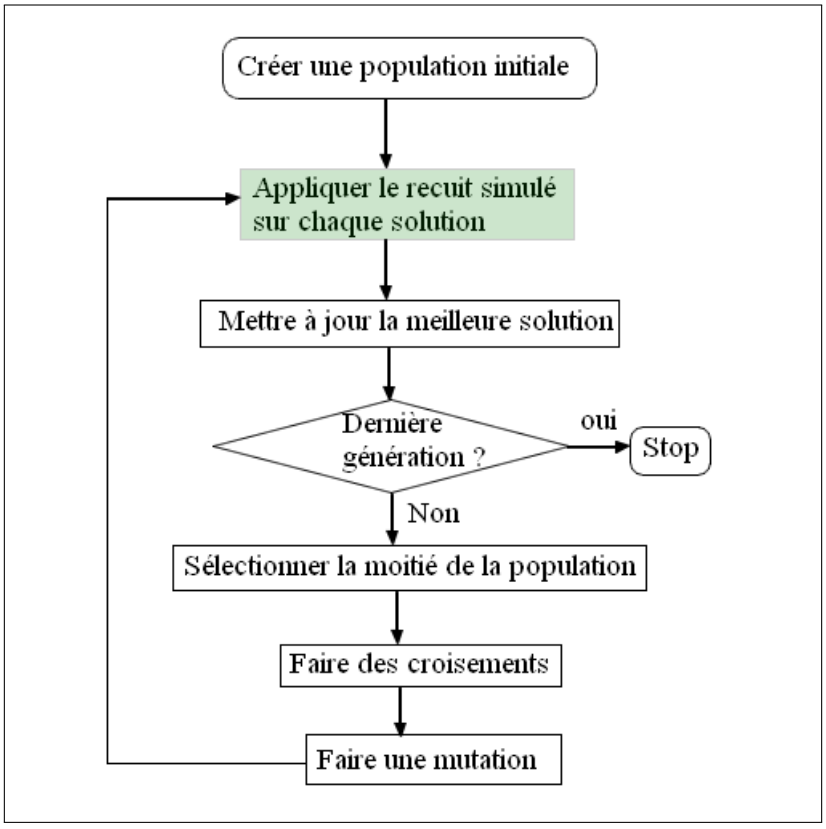


FIGURE 46 – Algorithme hybride génétique et recuit simulé

5.10.1 Résultats Numériques

Pour implémenter l'algorithme HGRS, nous avons utilisé les mêmes valeurs des paramètres qui sont utilisés dans l'algorithme génétique et dans l'algorithme du recuit simulé. Une comparaison des résultats de l'algorithme HGRS et des résultats optimaux trouvés par "ILOG CPLEX" est présentée dans le tableau 5.32.

Tableau 5.32: Résultats numériques de l'algorithme HGRS

Instances				CLPEX				HGRS				
Np	N	P_d	H	Résultat	Temps	Distance	NR	Résultat	Temps	Distance	NR	Gap
10	20	33%	2	23054	3.38 sec	15938	6	26181	10 sec	16129	10	0.135638
	21	53%		23874	3.12 sec	16731	6	28046	12 sec	16972	11	0.174751
	22	43%	3	28863	17.61 sec	17724	8	31515	10 sec	17473	14	0.0918823
	23	46%		23751	9.78 sec	18593	1	31400	11 sec	18359	13	0.32205
	24	70%		22529	195.91 sec	19310	3	30711	11 sec	18669	12	0.363176
	25	63%	4	21331	63.73 sec	20145	1	23291	17 sec	20274	3	0.091885
	26	36%		26962	225.02 sec	20831	3	32505	17 sec	20468	12	0.205586
	27			23107	2.11 sec	21951	1	25743	28 sec	21717	4	0.114078
	28	53%		24965	341.53 sec	22700	1	28749	17 sec	22700	6	0.151572
11	20	60%	2	18129	18.45 sec	16048	2	22007	13 sec	15959	6	0.213911
	21			20823	1393.86 sec	16682	4	26694	11 sec	16655	10	0.281948
	22	36%	3	24131	155.91 sec	18040	3	31611	10 sec	17556	14	0.309975
	23	69%		19490	19.78 sec	18360	1	27493	21 sec	18414	9	0.410621
	24	30%		32568	364.96 sec	19383	11	35403	15 sec	19324	16	0.0870486
	25	48%	4	22294	17.39 sec	20103	1	30521	18 sec	19457	11	0.369023
	26	39%		23801	38.39 sec	20646	2	24678	19 sec	20637	4	0.0368472
	27	45%		25252	1049.26 sec	22076	1	25295	18 sec	21195	4	0.00170284
Total				404924	3920.19	325261	55	481843	258	321958	159	3.36169

Les résultats présentés dans le tableau 5.32 montrent que l'écart relatif moyen entre les résultats de l'algorithme HGRS et les résultats optimaux trouvés par "ILOG CPLEX" est égal à 0.19775.

Des comparaisons entre l'algorithme HGRS, l'algorithme génétique, et l'algorithme du recuit simulée sont présentées dans les tableaux 5.33, 5.34, et 5.35.

Tableau 5.33: Comparaison des valeurs de la fonction objectif

Instances				HGRS	Génétique	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	26181	26305	27524
	21	53%		28046	28166	30826
	22	43%	3	31515	31705	33555
	23	46%		31400	31400	31986
	24	70%		30711	30800	33683

	25	63%	4	23291	23563	25892
	26	36%		32505	33588	35786
	27			25743	33613	34903
	28	53%		28749	31732	32260
11	20	60%	2	22007	25228	25362
	21			26694	26856	28018
	22	36%	3	31611	31687	31859
	23	69%		27493	30476	32340
	24	30%		35403	35447	37198
	25	48%	4	30521	30853	30757
	26	39%		24678	28902	30752
	27	45%		25295	29879	34155

Le tableau 5.33 montre que l'algorithme HGRS améliore les résultats de l'algorithme génétique et aussi ceux du recuit simulé.

Tableau 5.34: Comparaison des distances

Instances				HGRS	Génétique	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	16129	16253	17463
	21	53%		16972	17099	17756
	22	43%	3	17473	17676	19515
	23	46%		18359	18359	18931
	24	70%		18669	18765	18635
	25	63%	4	20274	20547	20871
	26	36%		20468	20544	20755
	27			21717	21583	21876
	28	53%		22700	22687	23174
11	20	60%	2	15959	16178	16312
	21			16655	16804	16972
	22	36%	3	17556	17624	17793
	23	69%		18414	18413	18274
	24	30%		19324	19381	20109
	25	48%	4	19457	19798	19696
	26	39%		20637	20865	20710
	27	45%		21195	21818	22087

Le tableau 5.34 montre que l'algorithme HGRS trouve les plus petites distances dans la plupart des cas, comparé à l'algorithme génétique et à l'algorithme du recuit simulé.

Tableau 5.35: Comparaison des nombres de remaniements

Instances				HGRS	Génétique	Recuit simulé
Np	N	P_d	H			
10	20	33%	2	10	10	10
	21	53%		11	11	13
	22	43%	3	14	14	14
	23	46%		13	13	13
	24	70%		12	12	15
	25	63%	4	3	3	5
	26	36%		12	13	15
	27			4	12	13
	28	53%		6	9	9
11	20	60%	2	6	9	9
	21			10	10	11
	22	36%	3	14	14	14
	23	69%		9	12	14
	24	30%		16	16	17
	25	48%	4	11	11	11
	26	39%		4	8	10
	27	45%		4	8	12

Le tableau 5.35 montre que l'algorithme HGRS trouve les plus petits nombres de remaniements dans la plupart des cas, comparé à l'algorithme génétique et à l'algorithme du recuit simulé.

Au vu de ces résultats, on peut dire que l'algorithme HGRS améliore l'algorithme génétique et l'algorithme du recuit simulé.

5.11 Comparaison des différents algorithmes

Dans cette section, nous comparons les résultats des différents algorithmes proposés dans ce chapitre. Pour ce faire, nous avons utilisé des instances de grandes dimensions. Dans un premier temps, nous nous focalisons sur les résultats des méta-heuristiques. Après cela, nous examinons ceux des algorithmes hybrides.

Les tableaux 5.36, 5.37, 5.38, et 5.39 confrontent les résultats des algorithmes suivants : recuit simulé, génétique, et colonie de fourmis.

Tableau 5.36: Comparaison des valeurs de la fonction objectif

Instances				Recuit	Génétique	Fourmis
Np	N	P_d	H			
	3020		12	5.28391e+08	5.13347e+08	4.98278e+08
	3521			6.15971e+08	5.98401e+08	5.80835e+08

1500	3722	49%	13	6.51228e+08	6.32667e+08	6.14066e+08
	4123			7.21272e+08	6.80164e+08	7.00764e+08
	4524			7.91542e+08	7.68988e+08	7.46382e+08
	4725		14	8.26652e+08	8.03049e+08	7.79473e+08
	5126			8.96878e+08	8.71304e+08	8.45694e+08
	5427			9.494e+08	8.95332e+08	9.22364e+08
	5828			1.0197e+09	9.90587e+08	9.61541e+08
2000	3020	50%	12	5.73766e+08	5.88825e+08	6.03866e+08
	3521			7.03985e+08	6.86431e+08	6.68844e+08
	3722		13	7.44282e+08	7.2573e+08	7.07136e+08
	4123			8.24344e+08	8.03804e+08	7.8321e+08
	4524			9.04616e+08	8.59479e+08	8.82081e+08
	4725		14	9.44681e+08	9.21167e+08	8.97566e+08
	5126			1.02497e+09	9.99467e+08	9.73865e+08
	5427			1.08509e+09	1.05805e+09	1.03093e+09
	5828			1.16541e+09	1.1363e+09	1.10719e+09
2500	3020	50%	12	6.7939e+08	6.64328e+08	6.49266e+08
	3521			7.9196e+08	7.74419e+08	7.56823e+08
	3722		13	8.37292e+08	8.18769e+08	8.00204e+08
	4123			9.27363e+08	9.06814e+08	8.86272e+08
	4524			1.01775e+09	9.95198e+08	9.72595e+08
	4725		14	1.0628e+09	1.01566e+09	1.03926e+09
	5126			1.15312e+09	1.12762e+09	1.10202e+09
	5427			1.22073e+09	1.1937e+09	1.16661e+09
	5828			1.31106e+09	1.25296e+09	1.28205e+09
3000	3020	50%	12	7.54873e+08	7.39842e+08	7.24737e+08
	3521			8.79942e+08	8.62398e+08	8.4486e+08
	3722		13	8.93223e+08	9.11817e+08	9.30385e+08
	4123			1.03043e+09	1.00988e+09	9.8931e+08
	4524			1.13081e+09	1.10826e+09	1.08567e+09
	4725		14	1.1809e+09	1.1338e+09	1.15738e+09
	5126			1.28133e+09	1.25577e+09	1.23015e+09
	5427			1.3564e+09	1.32932e+09	1.30221e+09
	5828			1.45678e+09	1.42773e+09	1.39866e+09

Les résultats du tableau 5.36 montrent que, même avec des instances de grandes tailles, c'est l'algorithme de colonie de fourmis qui trouve les meilleurs résultats, comparé au recuit simulé et à l'algorithme génétique.

Tableau 5.37: Comparaison des distances

Instances				Recuit	Génétique	Fourmis
Np	N	P_d	H			
1500	3020	49%	12	3.02002e+06	2.71808e+06	2.86906e+06
	3521			3.52101e+06	3.34503e+06	3.16891e+06
	3722		13	3.72201e+06	3.34981e+06	3.53598e+06
	4123			4.1231e+06	3.71073e+06	3.9169e+06

	4524			4.5241e+06	4.07165e+06	4.29787e+06
	4725		14	4.7251e+06	4.48882e+06	4.25255e+06
	5126			5.12603e+06	4.86979e+06	4.61348e+06
	5427			5.42705e+06	5.15566e+06	4.88438e+06
	5828			5.82805e+06	5.24528e+06	5.53661e+06
2000	3020	50%	12	3.02e+06	2.86909e+06	2.718e+06
	3521			3.52105e+06	3.16892e+06	3.34496e+06
	3722		13	3.72205e+06	3.34982e+06	3.53591e+06
	4123			4.12302e+06	3.91686e+06	3.71072e+06
	4524			4.52403e+06	4.07169e+06	4.29786e+06
	4725		14	4.72503e+06	4.25258e+06	4.4888e+06
	5126			5.12605e+06	4.8697e+06	4.61341e+06
	5427			5.42706e+06	4.88432e+06	5.1557e+06
	5828			5.82809e+06	5.24521e+06	5.53668e+06
2500	3020	50%	12	3.02002e+06	2.86908e+06	2.71802e+06
	3521			3.52109e+06	3.16896e+06	3.34502e+06
	3722		13	3.72209e+06	3.34986e+06	3.53597e+06
	4123			4.12305e+06	3.71074e+06	3.91686e+06
	4524			4.52408e+06	4.29782e+06	4.07168e+06
	4725		14	4.72507e+06	4.25256e+06	4.48883e+06
	5126			5.1261e+06	4.8698e+06	4.61345e+06
	5427			5.42706e+06	4.88432e+06	5.15571e+06
	5828			5.82803e+06	5.53665e+06	5.2453e+06
3000	3020	50%	12	3.02007e+06	2.718e+06	2.86901e+06
	3521			3.52107e+06	3.34496e+06	3.16896e+06
	3722		13	3.72209e+06	3.34982e+06	3.53596e+06
	4123			4.12301e+06	3.91689e+06	3.71079e+06
	4524			4.52409e+06	4.07169e+06	4.29781e+06
	4725		14	4.72505e+06	4.25257e+06	4.48875e+06
	5126			5.12608e+06	4.8698e+06	4.61345e+06
	5427			5.42701e+06	4.88436e+06	5.15567e+06
	5828			5.82809e+06	5.24521e+06	5.53662e+06

Le tableau 5.37 met en évidence que l'algorithme génétique trouve les plus petites distances, même si l'algorithme de colonie de fourmis minimise mieux la fonction objectif.

Tableau 5.38: Comparaison des nombres de remaniements

Instances				Recuit	Génétique	Fourmis
Np	N	P_d	H			
1500	3020	49%	12	127	127	126
	3521			120	112	104
	3722		13	132	121	109
	4123			159	114	140
	4524			145	125	113

	4725		14	152	135	111
	5126			137	131	107
	5427			154	122	145
	5828			171	156	133
2000	3020	50%	12	114	133	120
	3521			140	114	104
	3722		13	140	104	94
	4123			137	117	105
	4524			131	117	137
	4725		14	144	128	100
	5126			162	140	134
	5427			107	142	110
	5828			146	108	107
2500	3020	50%	12	119	116	114
	3521			114	108	101
	3722		13	121	114	108
	4123			142	137	114
	4524			126	117	115
	4725		14	106	105	113
	5126			139	112	105
	5427			127	120	110
	5828			109	101	111
3000	3020	50%	12	117	97	84
	3521			112	105	96
	3722		13	98	126	124
	4123			119	102	99
	4524			129	97	94
	4725		14	111	102	126
	5126			131	120	96
	5427			129	104	100
	5828			120	118	113

Le tableau 5.38 montre que l'algorithme de colonie de fourmis trouve les plus petits nombres de remaniements, comparé au recuit simulé et à l'algorithme génétique.

Tableau 5.39: Comparaison des durées d'exécution

Instances				Recuit	Génétique	Fourmis
Np	N	P_d	H			
1500	3020	49%	12	41 sec	11 min 54 sec	36 min 40 sec
	3521			50 sec	16 min 6 sec	53 min 20 sec
	3722		13	52 sec	16 min 48 sec	55 min
	4123			59 sec	20 min 18 sec	59 min 20 sec
	4524			1 min 9 sec	23 min 48 sec	1 h 12 min 40 sec
	4725		14	1 min 7 sec	25 min 48 sec	1 h 20 min
	5126			1 min 13 sec	28 min 42 sec	1 h 25 min 40 sec
	5427			2 min 13 sec	32 min 12 sec	1 h 27 min 20 sec

	5828			2 min 27 sec	35 min 42 sec	1 h 36 min 20 sec
2000	3020	50%	12	53 sec	14 min	58 min
	3521			1 min 2 sec	18 min 18 sec	1 h 07 min 20 sec
	3722		13	1 min 4 sec	20 min 54 sec	1 h 15 min 20 sec
	4123			1 min 13 sec	21 min 42 sec	1 h 28 min 40 sec
	4524			1 min 22 sec	26 min 24 sec	1 h 32 min 40 sec
	4725		14	1 min 24 sec	28 min 36 sec	1 h 47 min 20 sec
	5126			1 min 32 sec	29 min 42 sec	1 h 49 min 40 sec
	5427			1 min 40 sec	32 min 54 sec	1 h 53 min 40 sec
	5828			1 min 48 sec	37 min 48 sec	1 h 57 min 20 sec
2500	3020	50%	12	1 min 3 sec	15 min 24 sec	1 h 10 min 40 sec
	3521			1 min 15 sec	20 min 42 sec	1 h 17 min 20 sec
	3722		13	1 min 18 sec	21 min 18 sec	1 h 20 min
	4123			1 min 27 sec	27 min	1 h 28 min
	4524			1 min 37 sec	28 min 18 sec	1 h 34 min
	4725		14	1 min 39 sec	31 min 30 sec	1 h 41 min 20 sec
	5126			1 min 50 sec	34 min 18 sec	1 h 52 min 40 sec
	5427			1 min 58 sec	37 min 48 sec	1 h 55 min 3 sec
	5828			2 min 9 sec	42 min	1 h 58 min
3000	3020	50%	12	1 min 12 sec	16 min 48 sec	1 h 18 min 40 sec
	3521			1 min 26 sec	21 min	1 h 29 min 21 sec
	3722		13	1 min 30 sec	22 min 24 sec	1 h 34 min
	4123			1 min 40 sec	28 min	1 h 37 min 43 sec
	4524			1 min 53 sec	32 min	1 h 40 min 48 sec
	4725		14	1 min 55 sec	35 min 12 sec	1 h 44 min 14 sec
	5126			2 min 7 sec	39 min 12 sec	1 h 45 min 59 sec
	5427			2 min 16 sec	44 min 48 sec	1 h 51 min 19 sec
	5828			2 min 28 sec	47 min 36 sec	1 h 59 min 03 sec

Le tableau 5.39 illustre la rapidité du recuit simulé, par rapport à l'algorithme génétique et à l'algorithme de colonie de fourmis.

Tableau 5.40: Comparaison des valeurs de la fonction objectif

Instances				HCFG	HCFRS	HGRS
Np	N	P_d	H			
1500	3020	49%	12	4.5298e+08	4.68106e+08	4.83162e+08
	3521			5.28077e+08	5.45684e+08	5.63246e+08
	3722		13	5.98287e+08	5.76906e+08	5.95491e+08
	4123			6.68391e+08	6.8898e+08	6.5954e+08
	4524			6.7857e+08	7.0122e+08	7.23797e+08
	4725		14	7.08648e+08	7.32302e+08	7.55863e+08
	5126			8.00864e+08	7.94475e+08	8.20087e+08
	5427			8.83921e+08	8.41043e+08	8.68162e+08
	5828			8.74198e+08	9.0326e+08	9.32439e+08

2000	3020	50%	12	5.28478e+08	5.43578e+08	5.58682e+08
	3521			6.66044e+08	6.73649e+08	6.51239e+08
	3722		13	6.51318e+08	6.69949e+08	6.88538e+08
	4123			7.21402e+08	7.42032e+08	7.62643e+08
	4524			8.21708e+08	8.14315e+08	8.36905e+08
	4725		14	8.2676e+08	8.50377e+08	8.74003e+08
	5126			9.37014e+08	9.22612e+08	9.48257e+08
	5427			9.49598e+08	9.7673e+08	1.00387e+09
	5828			1.08987e+09	1.09904e+09	1.07813e+09
2500	3020	50%	12	6.44014e+08	6.79104e+08	6.34186e+08
	3521			7.04078e+08	7.2168e+08	7.39231e+08
	3722		13	7.44377e+08	7.62983e+08	7.81584e+08
	4123			8.24444e+08	8.45043e+08	8.65658e+08
	4524			9.34812e+08	9.27416e+08	9.49997e+08
	4725		14	9.44868e+08	9.68452e+08	9.92094e+08
	5126			1.02519e+09	1.05077e+09	1.07644e+09
	5427			1.08529e+09	1.11237e+09	1.13953e+09
	5828			1.16556e+09	1.19468e+09	1.2238e+09
3000	3020	50%	12	7.09473e+08	7.14605e+08	7.09665e+08
	3521			7.92058e+08	8.09634e+08	8.27249e+08
	3722		13	8.67455e+08	8.56053e+08	8.7463e+08
	4123			9.27528e+08	9.48145e+08	9.68698e+08
	4524			1.01792e+09	1.04049e+09	1.06311e+09
	4725		14	1.06297e+09	1.08654e+09	1.11021e+09
	5126			1.15334e+09	1.17899e+09	1.20458e+09
	5427			1.22092e+09	1.24806e+09	1.27514e+09
	5828			1.35128e+09	1.34036e+09	1.36952e+09

Le tableau 5.40 met en évidence que HCFG trouve des résultats meilleurs que ceux de HCFRS et HGRS, dans la plupart des cas.

Tableau 5.41: Comparaison des distances

Instances				HCFG	HCFRS	HGRS
Np	N	P_d	H			
1500	3020	49%	12	2.567e+06	2.41609e+06	2.265e+06
	3521			2.64083e+06	2.81683e+06	2.99294e+06
	3722		13	2.97763e+06	2.79158e+06	3.16379e+06
	4123			3.50456e+06	3.29844e+06	3.09228e+06
	4524			3.39307e+06	3.61926e+06	3.84546e+06
	4725		14	4.01631e+06	3.78006e+06	3.54382e+06
	5126			4.10081e+06	3.84452e+06	4.35715e+06
	5427			4.34163e+06	4.07028e+06	4.61299e+06
	5828			4.95384e+06	4.66243e+06	4.37102e+06
	3020		12	2.56708e+06	2.41603e+06	2.26508e+06
	3521			2.99293e+06	2.81681e+06	2.64078e+06

2000	3722	50%	13	2.79153e+06	2.97761e+06	3.16378e+06
	4123			3.50464e+06	3.29844e+06	3.09228e+06
	4524			3.61924e+06	3.39306e+06	3.84544e+06
	4725		14	4.01627e+06	3.78005e+06	3.54382e+06
	5126			4.10087e+06	3.84455e+06	4.35715e+06
	5427			4.07032e+06	4.34166e+06	4.61297e+06
	5828			4.9538e+06	4.66248e+06	4.37109e+06
2500	3020	50%	12	2.567e+06	2.41608e+06	2.26509e+06
	3521			2.64079e+06	2.81684e+06	2.99287e+06
	3722		13	3.16372e+06	2.97764e+06	2.79154e+06
	4123			3.50457e+06	3.29848e+06	3.09234e+06
	4524			3.61922e+06	3.39305e+06	3.84543e+06
	4725		14	3.54379e+06	3.78004e+06	4.0163e+06
	5126			4.35711e+06	4.10088e+06	3.84455e+06
	5427			4.61303e+06	4.34166e+06	4.0703e+06
	5828			4.37101e+06	4.6624e+06	4.95384e+06
3000	3020	50%	12	2.41609e+06	2.56704e+06	2.265e+06
	3521			2.64076e+06	2.81683e+06	2.99286e+06
	3722		13	2.97768e+06	2.7915e+06	3.16378e+06
	4123			3.50457e+06	3.29849e+06	3.09225e+06
	4524			3.84546e+06	3.61929e+06	3.39306e+06
	4725		14	4.01634e+06	3.78002e+06	3.54384e+06
	5126			4.35719e+06	4.10087e+06	3.84459e+06
	5427			4.07026e+06	4.34164e+06	4.61295e+06
	5828			4.66244e+06	4.37106e+06	4.95387e+06

Le tableau 5.41 montre que HGRS trouve les plus petites distances dans la plupart des cas, comparé à HCFG et HCFRS.

Tableau 5.42: Comparaison des nombres de remaniements

Instances				HCFG	HCFRS	HGRS
Np	N	P_d	H			
1500	3020	49%	12	91	121	123
	3521			102	103	107
	3722		13	108	107	113
	4123			114	135	113
	4524			97	105	111
	4725		14	88	92	101
	5126			104	95	126
	5427			102	94	107
	5828			95	103	107
	3020		12	90	97	92
	3521			100	102	96
	3722		13	91	93	105

2000	4123	50%		97	103	108
	4524			112	87	115
	4725		14	91	94	95
	5126			115	89	112
	5427			90	92	92
	5828			92	94	91
2500	3020	50%	12	90	99	88
	3521			96	98	98
	3722		13	96	99	101
	4123			78	85	111
	4524			115	78	99
	4725		14	88	97	97
	5126			84	94	92
	5427			83	97	91
	5828			81	88	93
3000	3020	50%	12	84	79	73
	3521			79	91	94
	3722		13	87	79	96
	4123			79	77	83
	4524			87	85	94
	4725		14	93	98	100
	5126			86	91	91
	5427			99	100	98
	5828			80	79	118

Le tableau 5.42 met en évidence que l'algorithme HCFG trouve les plus petits nombres de remaniements dans la plupart des cas, comparé à HCFRS et HGRS.

Tableau 5.43: Comparaison des durées d'exécution

Instances				HCFG	HCFRS	HGRS
Np	N	P_d	H			
1500	3020	49%	12	3 h 44 min 6 sec	1 h 27 min 13 sec	41 min 2 sec
	3521			3 h 24 min 54 sec	1 h 42 min 31 sec	58 min 55 sec
	3722		13	3 h 31 min 12 sec	2 h 15 min 09 sec	1 h 7 min 11 sec
	4123			3 h 2 min	2 h 22 min 22 sec	1 h 12 min 42 sec
	4524			3 h 34 min 12 sec	2 h 35 min 21 sec	1 h 34 min 36 sec
	4725		14	3 h 34 min 43 sec	2 h 10 min 17 sec	1 h 43 min 20 sec
	5126			3 h 08 min 14 sec	2 h 25 min 18 sec	1 h 18 min 23 sec
	5427			3 h 49 min 48 sec	2 h 19 min 32 sec	1 h 20 min 24 sec
	5828			3 h 21 min 13 sec	2 h 55 min 28 sec	1 h 21 min 19 sec
2000	3020	50%	12	3 h 6 min 10 sec	2 h 5 min 4 sec	56 min 9 sec
	3521			3 h 2 min 42 sec	2 h 12 min 15 sec	1 h 21 min 33 sec
	3722		13	3 h 50 min 7 sec	2 h 57 min 6 sec	1 h 44 min 26 sec
	4123			3 h 35 min 18 sec	2 h 43 min 33 sec	1 h 15 min 13 sec
	4524			3 h 24 min 36 sec	2 h 35 min 17 sec	1 h 48 min 51 sec
	4725			3 h 59 min 24 sec	2 h 37 min 45 sec	1 h 21 min 29 sec

	5126		14	3 h 18 min 18 sec	2 h 44 min 21 sec	1 h 57 min 33 sec
	5427			3 h 56 min 6 sec	2 h 57 min 34 sec	1 h 59 min 22 sec
	5828			3 h 40 min 12 sec	2 h 38 min 37 sec	1 h 48 min 53 sec
2500	3020	50%	12	3 h 18 min 36 sec	1 h 51 min 43 sec	58 min 23 sec
	3521			3 h 15 min 18 sec	2 h 5 min 11 sec	1 h 3 min 8 sec
	3722		13	3 h 22 min 42 sec	2 h 12 min 42 sec	1 h 31 min 34 sec
	4123			3 h 12 min	2 h 14 min 1 sec	1 h 34 min 10 sec
	4524			3 h 5 min 42 sec	2 h 27 min 39 sec	1 h 41 min 24 sec
	4725		14	3 h 43 min 30 sec	2 h 33 min 40 sec	1 h 54 min 53 sec
	5126			3 h 8 min 42 sec	2 h 51 min 29 sec	1 h 49 min 43 sec
	5427			3 h 40 min 12 sec	2 h 24 min 17 sec	1 h 50 min 37 sec
	5828			3 h 58 min	3 h 1 min 7 sec	1 h 59 min 36 sec
3000	3020	50%	12	3 h 31 min 12 sec	2 h 1 min 21 sec	1 h 13 min 44 sec
	3521			3 h 9 min	2 h 19 min 10 sec	1 h 27 min 3 sec
	3722		13	3 h 21 min 36 sec	2 h 21 min 43 sec	1 h 31 min 55 sec
	4123			3 h 12 min 10 sec	2 h 34 min 4 sec	1 h 41 min 9 sec
	4524			3 h 15 min	2 h 35 min 32 sec	1 h 15 min
	4725		14	3 h 49 min 48 sec	2 h 39 min 54 sec	1 h 34 min 28 sec
	5126			3 h 52 min 48 sec	2 h 33 min 45 sec	1 h 57 min 34 sec
	5427			3 h 43 min 12 sec	2 h 54 min 11 sec	1 h 53 min 21 sec
	5828			3 h 58 min 24 sec	2 h 59 min 14 sec	2 h 4 sec

Les résultats contenus dans le tableau 5.43 montrent que HGRS est plus rapide que HCFG et HCFRS.

5.12 Conclusion

Dans ce chapitre, nous avons traité le cas dynamique du problème de stockage de conteneurs dans un terminal portuaire. Pour ce faire, nous avons d'abord proposé un modèle mathématique qui reflète les contraintes réelles et qui minimise simultanément les nombres de remaniements et les distances totales à parcourir par les cavaliers gerbeurs, tout en regroupant les conteneurs par catégorie. Après cette modélisation mathématique, nous avons proposé six algorithmes pour la résolution numérique : un algorithme de colonie de fourmis, un algorithme génétique, un algorithme de recuit simulé, un algorithme hybride de colonie de fourmis et génétique, un algorithme hybride génétique et recuit simulé, et un algorithme hybride de colonie de fourmis et recuit simulé. Des simulations numériques et des comparaisons avec "ILOG CPLEX" ont montré que ces différents algorithmes fournissent de bon résultats, et que l'hybridation est un bon moyen pour améliorer les résultats des algorithmes.

Troisième partie

Conclusion générale

Conclusion générale

Dans ce travail, nous proposons des contributions scientifiques de résolution du problème de stockage de conteneurs dans un terminal portuaire. Ces contributions font suite à une étude bibliographique approfondie sur le sujet, et aussi à des visites effectuées dans des terminaux du port du Havre, qui ont permis de bien cerner le problème ainsi que les besoins des industriels portuaires. Le principal constat qui a été fait est la nécessité de systèmes efficaces d'aide à la décision qui permettraient aux planificateurs de gagner du temps tout en augmentant la productivité du port. Pour répondre à ce besoin, nous avons, dans un premier temps, effectué une étude analytique du problème dans laquelle nous étudions sa complexité, et proposons des modèles mathématiques qui ont pour objectif principal d'attribuer le meilleur emplacement à chaque conteneur. Ces modèles prennent en considération les contraintes physiques et logiques de stockage. Leurs principaux atouts sont :

- Le traitement simultané de tous les conteneurs (importés, exportés, ou en transbordement entre différents navires).
- La détermination d'un emplacement de stockage précis pour chaque conteneur.
- La prise en considération des propriétaires et des destinations des conteneurs, en regroupant les conteneurs qui seront chargés sur le même moyen de transport ou bien qui appartiennent à un même client.
- L'interdiction des remaniements, tout en prenant en considération le cas où il n'est pas possible de la respecter. Dans ce cas, nous minimisons les remaniements.
- Le traitement du cas du stockage dynamique.

Après la phase de modélisation mathématique, nous sommes passés à celle de la résolution numérique. Pour ce faire, nous avons commencé par utiliser un logiciel d'optimisation très connu, appelé "ILOG CPLEX" version 12.6. Les résultats des simulations numériques que nous avons effectuées avec ce logiciel, sur des instances de petites tailles, ont prouvé l'efficacité des modèles mathématiques. Cependant, puisque "ILOG CPLEX" n'arrivait pas à résoudre des instances de grandes tailles, nous avons implémenté un algorithme de Branch-and-Cut qui est aussi une méthode de résolution exacte. Des simulations numériques ont montré que cet algorithme est efficace et qu'il est capable de résoudre des instances qui n'ont pas pu être résolues avec "ILOG CPLEX". Le point faible de ces méthodes exactes est le fait qu'elles soient coûteuses en temps de calcul et en mémoire d'ordinateur. Pour contourner cette difficulté et accélérer la résolution des instances de grandes dimensions, nous avons proposé des algorithmes méta-heuristiques et des hybridations entre eux. Parmi ces algorithmes, il y en a un qui est séquentiel, c'est-à-dire qui utilise une seule solution initiale, et la modifie successivement via une méthode de recherche locale ; c'est le recuit simulé. Les autres algorithmes utilisent chacun une po-

pulation de solutions initiales, qu'ils améliorent en s'inspirant de phénomènes observés dans la nature. Ils sont au nombre de quatre : un algorithme de colonie d'abeilles, un algorithme de colonie de fourmis, et un algorithme génétique. Les algorithmes hybrides que nous proposons sont au nombre de trois, et sont constitués de ces combinaisons : colonie de fourmis et génétique, colonie de fourmis et recuit simulé, génétique et recuit simulé. Des tests de comparaison faits avec "ILOG CPLEX" sur de petites instances ont prouvé l'efficacité de ces algorithmes, et ont mis en évidence le fait que l'hybridation est un moyen efficace pour améliorer les performances d'un algorithme.

Publications dans des revues internationales avec comité de lecture

1) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba, *A Hybrid Ant Colony and Branch-and-Cut Algorithm to Solve the Container Stacking Problem at Seaport Terminal*, International Journal on Advances in Software, issn 1942-2628, vol. 7, no 3 & 4, pp. 567-580, 2014.
http ://www.iariajournals.org/software/soft_v7_n34_2014_paged.pdf.

2) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Hybrid algorithms to solve the container stacking problem at seaport*, Journal of mathematics, Statistics and Operations Research (JMSOR), vol. 2, no. 2, pp. 44-56, 2014.
https ://www.dropbox.com/s/2q4glqb8xwe2j3d/JMSOR_Vol.2_No.2_Paper_8.pdf?dl=0

4) Riadh Moussi, Jalel Euchi, Ndèye Fatma Ndiaye, Adnan Yassine. *A hybrid ant colony and simulated annealing algorithm to solve the container stacking problem at seaport terminal.* À paraître dans International journal of operational research.

5) Riadh Moussi, Ndèye Fatma Ndiaye, Adnan Yassine. *Hybrid Genetic Simulated Annealing Algorithm (HGSAA) to Solve Storage Container Problem in Port.* Intelligent Information and Database Systems Lecture Notes in Computer Science (Springer), Volume 7197, pp. 301-310, 2012.

Conférences internationales avec comité de lecture

1) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Comparison of efficient algorithms to solve the container stacking problem at seaport terminal.* The 5^{th} international conference on metaheuristics and nature inspired computing (META'2014). 27-31 Octobre 2014, MARRAKECH (Maroc).

2) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *A branch-and-cut algorithm to solve the container storage problem.* The ninth international conference on systems (ICONS 2014), IARIA. 23-27 Février 2014, Nice (France).

3) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *An ant colony algorithm to solve the storage container problem*. The 3^{rd} Annual International Conference on Computational Mathematics, Computational Geometry and Statistics (CMCGS 2014), pp. 28-33, 2014.
https ://www.dropbox.com/s/gh2k4kakjsm35rh/CMCGS_2014_Paper_%207.pdf

4) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *A Hybrid ant colony and genetic algorithm to solve the container stacking problem at seaport terminal*. The 3^{rd} international conference on advanced logistics and transports (ICALT'2014), IEEE. 1-3 Mai 2014, Hammamet (Tunisie).

5) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Algorithmes pour la résolution du problème de stockage de conteneurs dans un terminal portuaire*. Workshop optimisation de réseaux de transport et localisation d'activités qui s'inscrit dans le cadre du projet ORTRANS. 19-20 Novembre 2013. Dakar (Sénégal).

6) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Le problème de stockage de conteneurs dans un terminal portuaire*. $14^{ème}$ congrès annuel de la société française de recherche opérationnelle et d'aide à la décision (ROADEF2013). 13-15 Février 2013, Troyes (France).

7) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Algorithmes hybrides pour la résolution du problème de stockage de conteneurs dans un terminal portuaire*. $13^{ème}$ congrès annuel de la société française de recherche opérationnelle et d'aide à la décision (ROADEF2012). 11-13 Avril 2012, Angers (France).

8) Ndèye Fatma Ndiaye, Adnan Yassine, Ibrahima Diarrassouba. *Stockage de conteneurs dans un terminal portuaire*. Workshop de l'université de Mostaganem. 18-21 Décembre 2011, Algérie.

9) Moussi R, Ndiaye N F, Yassine, A. *Hybrid Genetic Simulated Annealing Algorithm (HGSAA) to Solve Storage Container Problem in Port*. The 4^{th} Asian Conference on Intelligent Information and Database Systems. 19-21 Mars, Kaohsiung, Taiwan.

10) Moussi R, Ndiaye N F, Yassine A. *A Genetic Algorithm to Solve Storage Container Problem in Port*. The International Maritime Transport and Logistics Conference "A Vision For Future Integration". 18-20 Décembre 2011, Alexandrie, Egypte.

Perspectives

Les méthodes de résolution que nous proposons dans cette thèse pour résoudre le problème de stockage de conteneurs dans un terminal portuaire sont très pertinentes. Cependant, ils existent plusieurs perspectives d'approfondissement de ce travail.

Dans un premier temps, il serait intéressant d'enrichir le modèle mathématique, en y ajoutant des contraintes opérationnelles telles que :

- La prise en considération d'imprévus comme des retards de conteneur (arrivée ou départ), des manques d'espace, etc.
- Séparer les conteneurs qui ont des éléments à risque tels que les conteneurs qui ont des denrées alimentaires et ceux qui ont des éléments toxiques.
- Estimer le nombre minimal de cavaliers gerbeurs nécessaire pour effectuer les opérations de stockage durant une période donnée.

Outres les enrichissements du modèle mathématique, des recherches peuvent être menées pour trouver d'autres algorithmes efficaces pour la résolution du problème de stockage de conteneurs. La difficulté majeure se trouve au niveau des méthodes exactes, dû au caractère NP_difficile du problème. Raison pour laquelle il serait intéressant de tester de nouvelles méthodes exactes, telle que le Branch-and-Price qui combine la méthode du Branch-and-Bound à celle du Branch-and-Cut.

Nous avons utilisé la méthode d'agrégation pour résoudre ce problème multi-objectif qu'est le problème de stockage de conteneurs dans un terminal portuaire. Il serait également intéressant de tester d'autres types d'approches tels que : les méthodes de sous-population, la méthode lexicographique, l'approche paréto, ou bien même des méthodes hybrides.

Ce domaine d'étude s'avère très pertinent car très utile surtout sur le plan économique. Vue la progression de la mondialisation et les facilités qu'apporte le conteneur sur le domaine du transport, des efforts intellectuels seront toujours utiles pour optimiser son utilisation en tenant compte de la modernisation des moyens de transport et des équipements de manutention. Nous espérons que cette thèse contribuera à l'allégement des travaux des gestionnaires d'espaces de stockage de conteneurs, et motivera d'autres chercheurs à s'intéresser à ce sujet.

Quatrième partie

Bibliographie

Bibliographie

[1] I. Ayachi, R. Kammarti, M. Ksouri, et P. Borne, *Harmonie search to solve the container storage problem with different container types*, International journal of computer applications, vol. 48, pp. 26-32, 2012.

[2] R. K. Ahuja, T. L. Magnanti, et J. B. Orlin, *Network flows theory algorithms and applications*, Prentice Hall, pp. 207-248, 1993.

[3] R. Akbari et K. Ziarati, *Multi-objective bee swarm optimization*, International journal of innovative computing, Information and control, vol. 8, 2012.

[4] U.S. Army Transportation Museum (n.d.). History & development of the container. (publication date unknown). Consulté le 29 June 2014 sur http ://www.transchool.eustis.army.mil/museum/CONEX.htm

[5] Brochure Hapag Lloyd Corporation, *Hamburg : Container specification*, Hamburg, 2002.

[6] A. E. Branch, *Elements of Shipping*, London : Routledge, 2007.

[7] A. Bruzzon et R. Signorile, *Simulation and genetic algorithms for ship planning and shipyard layout*, Simulation, vol. 72, pp. 74-83, 1998.

[8] A. Bruzzone et R. Signorile, *Simulation and genetic algorithms for ship planning and ship yard layout*, Simulation, vol. 71, pp. 74-83, 1998.

[9] B. Borgman, E. V. Asperen, et R. Dekker, *Online rules for container stacking*, OR Spectrum, vol. 32, pp. 687-716, 2010.

[10] B. Bullnheimer, R. F. Hartl, et C. Strauss, *A new rank-based version of the Ant System : A computational study*, Central european journal for operations research and economics, vol. 7, pp. 25-38, 1999.

[11] E. Bonabeau, M. Dorigo, et G. Theraulaz, *Swarm Intelligence : from Natural to Artificial Systems*, Oxford University Press, New York, 1999.

[12] F. Belmecheri-Yalaoui, F. Yalaoui, et L. Amodeo, *Multi-objective Ant Colony Optimization Method to Solve Container Terminal Problem*, Applications of Multi-Criteria and Game Theory Approaches, part I, pp. 107-122, 2014.

[13] F. Bonomo, S. Mattia, et G. Oriolo, *Bounded coloring of co-comparability graphs and the pickup and delivery tour combination problem*, Theoretical Computer Science, vol. 412, pp. 6261-6268, 2011, Elsevier.

[14] F. Broeze, *The globalisation of the oceans : Containerisation from the 1950s to the present*, St. Johns, NF, Canada : International Maritime Economic History Association, 2002.

[15] H. L. Bodlaender et K. Jansen, *Restrictions of graph partition problems*, Part I, Theoretical Computer Science, vol. 148, pp. 93-109, 1995, Elsevier.

[16] H. C. Brookfield, *Boxes, ports, and places without ports*, In Doyle, B.S. & Hilling, D. (Eds.), Seaport systems and spatial change, pp. 61-79, Chichester, UK : Wiley, 1984.

[17] R. Bellman, *Dynamic programming*, Princeton university press, Princeton, NJ, 1957.

[18] R. D. Brown, *The port of London*, Lavenham, UK : Lavenham Press, 1978.

[19] S. Boyd, L. Xiao, et A. Mutapcic, *Subgradient methods*, Notes for EE392o, Stanford university, Autumn, 2003.

[20] B. Castilho et C. F. Daganzo, *Handling strategies for import containers at marine terminals*, Transportation Research Part B : Methodological, vol. 27, pp. 151-166, 1993.

[21] B. J. Cudahy, *The container revolution : Malcolm McLean's 1956 innovation goes global*, TR News 246, pp. 5-9, 2006.

[22] B. J. Cudahy, *Box boats : How container ships changed the world*, New York : Fordham UP, 2006.

[23] C. A. Coello, D. A. Van Veldhuizen, et G. B. Laumont, *Evolutionnary algorithms for solving multi-objective problem*, Kluwers academic publisher, New York, 2002.

[24] C. Y. Chu, et W. C. Huang, *Determining container terminal capacity on the basis of an adopted yard handling*, Transport Reviews : A Transnational Transdisciplinary Journal, vol. 25, pp. 181-199, 2005.

[25] H. J. Carlo, I. F. A. Vis, et K. J. Roodbergen, *Storage yard operations in container terminals : Literature overview, trends, and research directions*, European Journal of Operational Research, 2013.

[26] J. Culioli, *Introduction à l'optimisation*, Ellipses, 2^e édition, 2012.

[27] M. Clerk et J. Kennedy, *The particle swarm explosion, stability, and convergence in a multidimensional complex space*, IEEE transactions on evolutionary computation, vol. 6, pp. 58-73, 2002.

[28] S. Camazine, J. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula et E. Bonabeau, *Self-Organization in Biological Systems*, Princeton : Princeton University Press, 2003.

[29] S. Cohen, *Boom boxes : Shipping containers and terrorists*, Berkeley Roundtable on the International Economy (BRIE), University of California, Berkeley, Consulté le 30 juin 2014 sur http ://brie.berkeley.edu/publications/RP7.pdf

[30] T. Chen, *Yard operations in the container terminal-a study in the "unproductive" moves*, Marit Policy Manage, vol. 26, pp. 27-38, 1999.

[31] Unisys Corp, *Apec secure trade project final report*, Consulté le 30 juin 2014 sur http ://www.apec-tptwg.org.cn/new/Projects/APEC%20Final%20Report%20vFINAL.pdf, juin 2014.

[32] VeriTainer Corporation, *Trust, but verify*, Consulté le 30 juin 2014 sur http ://www.veritainer.com/technology.html, juin 2014

[33] B. Dushnik et E. W. Miller, *Partially ordered sets*, American Journal of Mathematics (The Johns Hopkins University Press), vol. 63, pp. 600-610, 1941.

[34] J.-L. Deneubourg, S. Aron, S. Goss, et J.-M. Pasteels. *The self-organizing exploratory pattern of the Argentine ant*, Journal of insect behavior, vol. 3, pp. 159-168, 990.

[35] M. Dorigo, *Optimization, learning and natural algorithms* (in Italian), PhD thesis, Dipartemento di Elottronica, Politechnico di Milano, Italy, pp. 140, 1992.

[36] M. Dorigo, V. Maniezzo, et A. Colorni. *The ant system : optimization by a colony of cooperating agents*, IEEE transactions on systems, man, and cybernetics - part B, vol. 26, pp. 29-41, 1996.

[37] M. Dorigo et L. M. Gambardella, *Ant colony system : a cooperative learning approach to the traveling salesman problem*, IEEE Transaction on evolutionary computation, vol. 1, pp. 53-66, 1997.

[38] M. Dorigo et L. M. Gambardella, *Ant colonies for the traveling salesman problem*, BioSystems, vol. 43, pp. 73-81, 1997.

[39] M. B. Duinkerken, J. J. M. Evers, et J. A. Ottjes, *A Simulation Model For Integrating Quay Transport And Stacking Policies On Automated Container Terminals*, Proceedings of the 15 th European Simulation Multiconference, Prague, 2001.

[40] R. Dekker, P. Voogd, et E. V. Asperen, *Advanced methods for container stacking*, Container Terminals and Cargo Systems, pp. 131-154, 2007.

[41] Erie et P. Stephen, *Globalizing L.A. : Trade, infrastructure, and regional development*, Stanford : Stanford UP, 2004.

[42] J. Frittelli, *Port and maritime security : Background and issues*, Military Technology, vol. 30, pp. 88-94. Consulté le 30 juin 2014 sur http ://www.fas.org/sgp/crs/homesec/RL31733.pdf, juin 2014

[43] Hans-Otto Günther et Kap-Haw Kim, *Container terminals and terminal operations*, OR Spectrum, vol. 28, pp. 437-445, 2006.

[44] L. M. Gambardella et M. Dorigo, *Solving symmetric and asymmetric TSPs by ant colonies.* In proceedings of the 1996 IEEE International conference on evolutionary computation (ICEC'96), pp. 622-627, IEEE press, Piscataway, NJ, 1996.

[45] Mitsuo Gen et Runwei Cheng, *Genetic Algorithms and Engineering Optimization*, Industrial Engineering / Manufacturing, 2007.

[46] R. Goss, *Economic policies and seaports : 4 strategies for port authorities*, Maritime Policy and Management, vol. 17, pp. 273-87, 1990.

[47] S. Goss, S. Aron, J. L. Deneubourg, et J. M. Pasteels, *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, vol. 76, pp. 579-581, 1989.

[48] Z. W. Geem, J. H. Kim, et G. V. Loganathan, *A new heuristic optimization algorithm : Harmony search*, Simulation, vol. 76, pp. 60-68, 2001.

[49] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Computers, 1975.

[50] M. Held, P. Wolfe, et H. P. Crowder, *Validation en subgradient optimization*, Mathematical programming, vol. 6, pp. 62-88, 1973.

[51] M. Huybrechts, H. Meersman, E. Van de Voorde, E. Van Hooydunk, A. Verbeke, et W. Winkelmans, *Port competitiveness : An economic and legal analysis of the factors determining the competiveness of seaports*, pp. 10-13. Antwerp : De Boeck Limited, 2002

[52] P. Hall, R. J. McCalla, C. Comtois, et B. Slack, *Integrating seaports and corridors*, Farnham, Surrey : Ashgate, 2011.

[53] X. Hu, R. C. Eberhart, et Y. Shi, *Particle swarm with extended memory for multiobjective optimization*, proceeding of the 2003 IEEE swarm intelligent symposium, pp. 193-197, 2003.

[54] J. Jones, *Active non-intrusive technologies for homeland defense*, Consulté le 30 juin 2014 sur http ://www.inl.gov/technicalpublications/Documents/2585364.pdf, juin 2014

[55] K. Jansen, *The mutual exclusion scheduling problem for permutation and comparability graphs*, Information and Computation, vol. 180, pp. 145-151, 2003, Springer.

[56] X. Jiang, L. H. Lee, Y. Han, et K. C. Tan, *A container yard strategy for improving land utilization and operation efficiency in a transshipment hub port*, European journal of operational research, vol. 221, pp. 64-73

[57] E. Kozan et P. Preston, *Genetic algorithms to schedule container transfers at mutimodal terminals*, International transactions in operational research, vol. 6, pp. 311-329, 1999.

[58] E. Kozan et P. Preston, *Mathematical modelling of container trasfers and storage locations at seaport terminals*, OR Spectrum, vol. 28, pp. 519-537, 2006.

[59] K. H. Kim, *Evaluation of the number of rehandles in container yards*, Computers & Industrial Engineering, vol. 32, pp. 701-711, 1997.

[60] K. H. Kim et J. W. Bae, *Re-marshaling export containers in port container terminals*, Computers & Industrial Engineering, vol. 35, pp. 655-658, 1998.

[61] K. H. Kim et K. T. Park, *A note on a dynamic space-allocation method for outbound containers*, European journal of operational research, vol. 148, pp. 92-101, 2003.

[62] K. H. Kim, Y. M. Park, et K. Ryu, *Deriving decision rules to locate export containers in containers yards*, European journal of operational research, vol. 124, pp. 89-101, 2000.

[63] K. H. Kim et K. Y. Kim, *Optimal price schedules for storage of inbound containers*, Transportation research, Part B, vol. 41, pp. 892-905, 2007.

[64] L. P. Ku, L. H. Lee, et E. P. Chew, *An optimization framework for yard planning in a container terminal : case with automated rail-mouted gantry cranes*, OR Spectrum, vol. 32, pp. 519-541, 2010.

[65] S. Kirkpatrick, C. D. Gelatt Jr, et M. P. Vecchi, *Optimization by simulated annealing*, vol. 220, pp. 671-680, 1983.

[66] Lloyd's, *Containerisation International Yearbook 2012, incl. CD-ROM.*

[67] C.-C. Lai, C.-H. Wu, et M.-C. Tsai, *Feature selection using particle swarm optimisation application in spam filtering*, International journal of innovatie computing, information and control, vol. 5, pp. 423-432, 2009.

[68] C. Liu, et Y. Wang, *A new evolutionary algorithm for multi-objective optimization problems*, ICIC express letters, vol. 1, pp. 93-98, 2007.

[69] D. Liu, C. K. Tan, C. K. Go, et W. K. Ho, *A multiobjective memetic algorithm based on particle swarm optimization*, IEEE transactions on systems, man, and cybernetics - Part B : cybernetics, vol. 37, pp. 42-50, 2007.

[70] J. Lam, S. Lee, et W. Y. Yap, *Dynamics of liner shipping network and port connectivity in supply chain systems : analysis on East Asia*, Journal of Transport Geography, vol. 19, pp. 1272-1281, 2011.

[71] K. S. Lee et Z. W. Geem, *A new meta-heuristic algorithm for continuous engineering optimization : harmony search theory and practice*, Computer methods in applied mechanics and engineering, vol. 194, pp. 3902-3933, 2005.

[72] L. H. Lee, E. P. Chew, K. C. Tan, et Y. Han, *An optimization model for storage yard management in transshipment hub*, OR Spectrum, vol. 28, pp. 539-561, 2006.

[73] M. Levinson, *The box : How the shipping container made the world smaller and the world economy bigger*, Princeton : Princeton UP, 2006.

[74] Y. Lee et N.-Y. Hsu, *An optimization model for the container pre-marshaling problem*. Computer and Operations research, vol. 34, pp. 3295-3313, 2007.

[75] Y. Lee et N. Hsu, *An optimization model for the container pre-marshaling problem*, Computer & operations research, vol. 34, pp. 3295-3313, 2007.

[76] J. Mangan, C. Lalwani, et B. Fynes, *Port-centric logistics*, 2008.

[77] J. Mangan, *Global Logistics and Supply Chain Management. Chichester : Wiley*, 2012.

[78] K. G. Murty, Y. Wan, J. Liu, M. M. Tseng, E. Leung, K. Lai, W. Herman, et C. Chiu, *Hongkong international terminal gains elastic capacity using a data-intensive decision support system*, Iterfaces, vol. 35, pp. 61-75, 2005.

[79] Riadh Moussi, *Les techniques modernes de l'optimisation combinatoire pour la résolution de problèmes d'optimisation dans les terminaux portuaires à conteneurs*, Thèse soutenue publiquement le 04/07/2012 à l'université du Havre.

[80] R. J. McCalla, *From 'anyport' to 'superterminal' : Conceptual perspectives on containerization and port Infrastructures*, In Pinder, D. & Slack, B. (Eds.) Shipping and ports in the twenty-first century : Globalization, technical change and the environment, London : Routledge, pp. 125-142, 2004.

[81] Y. Ma et K. H. Kim, *A comparative Analysis : Various storage rules in container yards and their perfermances*, Industrial engineering & Management systems, vol. 11, pp. 276-287, 2012.

[82] T. E. Notteboom, *Concentration and the formation of multi-port gateway regions in the European container port system : an update*, Journal of Transport Geography, vol. 18, pp. 567-583, 2010.

[83] E. Ozcan et C. K. Mohan, *Particle swarm optimization : surfing the waves*, congress on evolutionary computation, Washington D.C., USA, pp. 1939-1944, 1999.

[84] B. T. Poljak, *A general method of solving extremum problems*, Soviet mathematics doklady, vol. 8, pp. 593-597, 1967.

[85] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, et M. Zaidi, *The bees algorithm - A novel tool for complex optimization problems*, Manufacturing engineering center, Cardiff university : Cardiff, UK, 2005.

[86] K. E. Parsopoulos et M. N. Vrahatis, *Particle swarm optimisation method in multiobjective problems*, Proceeding of the 2002 ACM symposium on applied computing, pp. 603-607, 2002.

[87] M. E. Petering et K. G. Murty, *Simulation analysis of algorithms for container storage and yard crane scheduling at a container terminal*, Proceedings of the second international intelligent logistics systems conference, Brisbane, Australia, pp. 1-19, 2006.

[88] S. J. Pettit et A. K. C. Beresford, *From gateways to logistical hubs*. Maritime Policy & Management, vol. 36, pp. 253-267.

[89] SAIC, *From science to solutions*, Consulté le 30 juin 2014 sur http ://www.saic.com/, juin 2014

[90] B. Slack, *Corporate realignment and the global imperatives of container shipping*, In Pinder, D. & Slack, B. (Eds.) Shipping and ports in the twenty-first century : Globalization, technical change and the environment, London : Routledge, pp. 25-39, 2004.

[91] D. Sculli et C. F. Hui, *Three dimensional stacking of containers*, Omega, vol. 16, pp. 585-594, 1988.

[92] D. Steeken, S. Vob, et R. Stahlbock, *Container terminal operation and operations research–a classification and literature review*, OR Spectrum, vol. 26, pp. 309-332, 2005.

[93] J. Stowsky, *Harnessing a trojan horse : Aligning security investments with commercial trajectories in cargo container shipping*, Consulté le 30 juin 2014 sur http ://brie.berkeley.edu/publications/stowsky%20port%20security.pdf

[94] M. Stopford, *Maritime Economics*, New York : Routledge, 2009.

[95] N. Z. Shor, *Minimization methods for non-differentiable functions*, Springer series in computationnal mathematics, Springer, 1985.

[96] Rapidscan Systems, *Neutron technology*, Consulté le 30 juin 2014 sur http ://www.rapiscansystems.com/neutron.html, juin 2014

[97] R. Starr, *The rise and fall of New York City*, New York : Basic Books, 1985.

[98] T. D. Seeley, *The Wisdom of the Hive : The Social Physiology of Honey Bee Colonies*, Massachusetts : Harvard University Press, Cambridge, 1996.

[99] T. Stützle et H. H. Hoos, *The MAX-MIN Ant System and local search for the traveling salesman problem*, In T. Bäck, Z. Michalewicz, and X. Yao editors, Proceeding of the 1997 IEEE international conference on evolutionary computation (ICEC'97), pp. 309-314, IEEE Press, Piscataway, NJ, 1997.

[100] T. Stützle, *Local search algorithms for combinatorial problems : analysis, improvemevents, and new applications*, Infix, Sankt Augustin, Germany, 1999.

[101] T. Stützle et H. H. Hoos, *MAX-MIN Ant System*, Futur generation computer systems, vol. 16, pp. 889-914, 2000.

[102] Y. A. Saanen et R. Dekker, *Intelligent stacking as way out of congested yards ? part 1*, Port Technol Int, vol. 31, pp. 87-92, 2007.

[103] Y. A. Saanen et R. Dekker, *Intelligent stacking as way out of congested yards ? part 1*, Port Technol Int, vol. 32, pp. 80-85, 2007.

[104] M. Taleb-Ibrahimi, B. D. Castilho, et C. F. Daganzo, *Storage space vs handling work in container terminals*, Transportation Research Part B : Methodological, vol. 27, pp. 13-32, 1993.

[105] UNCTAD, Review of maritime transport, 2005 : United Nations, 2006.

[106] F. K. Von, *Bees : Their Vision, Chemical Senses and Language*, (Revised edn) Cornell University Press, N.Y., Ithaca, 1976.

[107] Van De Voort, M. O'Brien, K. Rahman, et R. Valeri, *"Seacurity" : Improving the security of the global sea-container shipping system*, Consulté le 30 juin 2014 sur http ://www.rand.org/pubs/monograph_reports/2005/MR1695.pdf

[108] www.container-transportation.com/container-terminal.html, consulté le 04 Juillet 2014.

[109] J. Wiese, L. Suhl, et N. Kliewer, *Planning container terminal layouts considering equipment types and storage block design*, Handbook of Terminal Planning Operations Research/Computer Science Interfaces Series, vol. 49, pp. 219-245, 2011.

[110] M. Wang, *The rise of container transport in Asia*, In : T. W. Lee, Culliane K (eds) World shipping and port development. Palgrave, Basingstoke, pp. 10-35, 2005.

[111] M. Yu et X. Qi, *Storage space allocation models for inbound containers in an automatic container terminal*, European journal of operational research, vol. 226, pp. 32-45.

[112] C. Zhang, J. Lui, Y. W. Wan, K. G. Murty, et R. Linn, *Storage space allocation in container terminals*, Transportation Research Part B : Methodological, vol. 37, pp. 883-903, 2003.

[113] X. Zhang, H. Meng, et L. Jiao, *Intelligent particle swarm optimization in multiobjective optimization*, Congree on evolutionary computation, pp. 714-719, 2005.